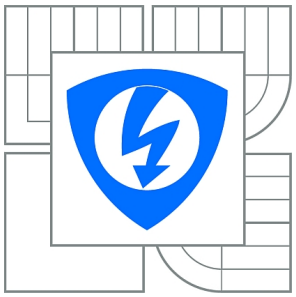
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO DOPLNĚNÍ CHYBĚJÍCÍCH ČÁSTÍ OBRAZU

IMAGE INPAINTING METHODS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

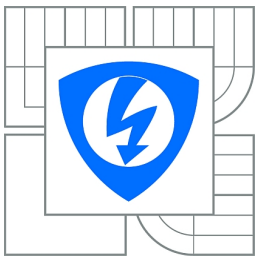
AUTOR PRÁCE — JAN KOVACS

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK PRŮŠA

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jan Kovacs	***ID:***	106552
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Metody pro doplnění chybějících částí obrazu

POKYNY PRO VYPRACOVÁNÍ:

Z dostupné literatury zpracujte přehled a zhodnocení moderních metod pro doplnění chybějících částí obrazu. Vybranou metodu implementujte jako funkci v prostředí MATLAB. V prostředí MATLAB dále navrhněte a realizujte GUI s dostatečnou úrovní uživatelské interakce. Bude možné načítat různé obrazy, nastavovat vnitřní parametry algoritmu pro doplnění obrazu a definovat samotné části obrazu, které budou považovány za chybějící. Na závěr zhodnoťte účinnost a výpočetní náročnost implementovaného algoritmu.

DOPORUČENÁ LITERATURA:

[1] Grochol, J.: Doplňování chybějících částí obrazu. Diplomová práce. Brno, MUNI, 2009. 50 s. Vedoucí diplomové práce: RNDr. Vít Kovalčík Ph.D.
[2] Bertalmio, M.: Sapiro, G.; Ballester, C.; aj.: Image Inpainting. SIGGRAPH, 2000. s. 417–424.
[3] Cai J., Chan R., Shen Z.: A framelet-based image inpainting algorithm, Applied and Computational Harmonic Analysis, 24 (2008), 131-149.

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Ing. Zdeněk Průša
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá zpracováním přehledu moderních metod pro automatické doplnění chybějících částí obrazu. V teoretické části této práce je vybráno a popsáno několik nejznámějších metod. Každá z vybraných metod je nejdříve uvedena, poté je popsán její algoritmus a nakonec je zhodnocena za pomoci informací, nabytých z dostupné literatury. Mezi metody, které byly vybrány a následně popsány v této práci patří Image Inpainting, Fragment-Based Image Completion, Exemplar-Based Image Inpainting, Gradient-Based Image Completion by Solving Poisson Equation a nakonec Inpainting by Flexible Haar-Wavelet Shrinkage.

V praktické části bakalářské práce byl vybrán algoritmus A Framelet-Based Image Inpainting, který byl naprogramován a implementován v programovém prostředí MATLAB. Pro tento algoritmus bylo také naprogramováno vlastní funkční řešení Framelet transformace. Dále bylo vytvořeno GUI, které poskytuje možnost uživatelské interakce. Toto v prostředí MATLAB realizované GUI umožňuje jednoduše spravovat vstupy a parametry algoritmu a pracovat s jeho výstupy. Uživatel je vždy informován o aktuálním stavu výpočtu a je mu zobrazen aktuální výsledek doplnění obrazu. Navíc byl pro GUI vytvořen nástroj, který poskytuje uživateli možnost definovat pomocí myši oblasti, jež mají být doplněny. Nakonec byly zhodnoceny výsledky implementovaného algoritmu jak při použití Framelet transformace, tak při použití Contoulet transformace.

KLÍČOVÁ SLOVA

metody pro doplnění obrazu, doplnění obrazu, parciální diferenciální rovnice, syntéza textury, přístup kopírování bloků, Framelet transformace, měkké prahování

ABSTRACT

This thesis deals with an overview of modern Image Inpainting Methods. There are several best-known methods selected and described in the theoretical part of this work. Each of the selected methods is described and evaluated according to the informations available in literature. Among the methods that were selected and subsequently described in this work are Image Inpainting, Fragment-Based Image Completion, Exemplar-Based Image Inpainting, Gradient-Based Image Completion by Solving Poisson Equation and Inpainting by Flexible Haar-Wavelet Shrinkage.

The MATLAB implementation of the Framelet-Based Image Inpainting algorithm forms practical part of the thesis. The Framelet transform was created for the purposes of the algorithm. The user interaction provides GUI, which was also implemented in MATLAB. The GUI allows setting input images, algorithm parameters and interaction with the output. The user is always informed about the current state of the computation, and the current result of image completion is shown to him. Moreover, it was created a tool that allows the user to define the areas to be supplemented, using the mouse. Finally, the algorithm performance is evaluated and compared using both Framelet and Contourlet transform.

KEYWORDS

Image Inpainting Methods, image completion, Partial Differential Equations, texture synthesis, Patch-Based approach, Framelet transform, Soft Threshold

KOVACS, J. *Metody pro doplnění chybějících částí obrazu*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 56 s. Vedoucí práce byl Ing. Zdeněk Průša

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Metody pro doplnění chybějících částí obrazu“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu své bakalářské práce panu Ing. Zdeňkovi Průšovi za odborné vedení, dobré připomínky, cenné rady, konzultace a podnětné návrhy k práci.
Dále děkuji své mamince za to, že mě podporovala ve studiu. Nakonec děkuji své sestře Mirce Sárové za pomoc s korekturou.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Myšlenka automatického digitálního doplňování obrazu byla poprvé uvedena v roce 2000 autory Bertalmio a spol. v revolučním článku Image Inpainting [4]. Jejich cílem bylo pokusit se vytvořit automatický algoritmus, jež bude postupovat stejnou metodou, jako profesionální restaurátoři. Vydání tohoto díla zažehlo velkou vlnu zájmu o oblast digitálního doplňování obrazu.

Do dnešního dne bylo vyvinuto několik metod pro doplnění obrazu založených na různých principech. Některé se dočkaly zavedení do praxe, dobrým příkladem je nástroj PatchMatch [3] vyvinutý pro aplikaci Adobe Photoshop. Jedná se o nástroj umožňující provádět s obrazem různé operace, včetně nahrazení jeho částí.

Jedním z cílů této bakalářské práce bylo vytvoření přehledu o současných metodách pro doplnění obrazu. První kapitola se věnuje popsání a vysvětlení některých použitých pojmů. Následuje kapitola přehledu metod pro doplnění obrazu.

Praktické části se věnuje třetí kapitola. Dle zadání byla vybrána metoda, která byla následně implementována v prostředí MATLAB. V rámci této práce bylo také naprogramováno vlastní řešení Framelet transformace. Dále bylo vytvořeno přehledné GUI umožňující uživatelskou interakci.

V poslední sekci třetí kapitoly jsou popsány výsledky doplnění obrazu pomocí implementovaného algoritmu. Jednak se zde popisuje vliv parametrů výpočtu na výsledek, jednak je zde zobrazen a popsán průběh doplnění obrazu. Průběžně jsou porovnávány výsledky při použití obou typů transformace a nakonec je zhodnocena účinnost implementovaného algoritmu.

1 ZÁKLADNÍ POJMY

V první řadě je potřeba vysvětlit si některé pojmy použité v textu a matematická označení jednotlivých funkcí, použitých v této práci.

1.1 Vstupní data

Diskrétní obraz, jak je popsán v [12], lze chápat jako zobrazení

$$f : \{1, 2, ..., N_x\} \times \{1, 2, ..., N_y\} \longrightarrow H_1 \times H_2 \times ... \times H_n, \tag{1.1}$$

kde H_m vyjadřuje množinu hodnot, které může nabývat m-tá složka modelu (kanál) pro $m = 1, .., n$. Obraz lze tedy chápat jako n matic o rozměru $N_x \times N_y$.

Digitální obraz si lze představit jako diskrétní obraz dle [12], jehož množiny hodnot jednotlivých složek jsou kvantovány na určitý počet vzorků. Například množina hodnot m-tého kanálu nabývá $H_m = \{0, 1, 2, ..., 255\}$ v případě osmibitového kvantování.

Vstupní obraz I je digitální obraz. Vstupní obraz může být:

1. V odstínech šedi (monochromatický). Obraz je tvořen pouze jedinou maticí osmibitových hodnot udávajících jas.
2. Barevný. V případě této práce se bude uvažovat převážně barevný model RGB, jehož obraz bude složen ze tří matic osmibitových hodnot, přičemž každá matice udává hodnoty jedné z barevných složek.

Vstupní maska, nebo také *maska* Ω, je podle definice digitální obraz stejné velikosti jako vstupní obraz. Obsahuje však pouze jediný kanál, jehož množina hodnot H obsahuje pouze hodnoty $H = \{0, 1\}$. Tato binární maska slouží k jednoznačnému určení oblasti Ω, tedy oblasti, která má být nahrazena. Body masky s hodnotou logická 1 určují oblast Ω, čili nahrazovanou oblast. Body masky s hodnotou 0 značí, které body vstupního obrazu budou zachovány [11].

Vynásobením vstupního obrazu I a vstupní masky se získá obraz obsahující známou oblast Φ, neznámou oblast Ω a lze určit jejich hranici $\partial\Omega$.

1.2 Matematické pojmy

Gradient funkce

Gradient funkce $f(x, y)$ udává směr největšího růstu funkce $f(x, y)$. Při formálním zápisu se jako gradient obvykle použije nabla operátor ∇ , někdy lze vidět i písemné označení grad. Směr takového vektoru lze určit dle [18] pro spojité funkce vzorcem:

$$\nabla f(x, y) = \text{grad} f(x, y) = \left(\frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y} \right), \tag{1.2}$$

velikost gradientu je potom délka vektoru:

$$| \nabla f(x, y) | = \sqrt{\left(\frac{\partial f(x, y)}{\partial x} \right)^2 + \left(\frac{\partial f(x, y)}{\partial y} \right)^2}. \tag{1.3}$$

V diskrétním obraze se gradient odhaduje.

Divergence

Jak je napsáno v [9], divergence vrací skalární hodnoty, vyjadřující chování okolních vektorů. Při kladné divergenci směřují vektory z daného místa ven (tzv. zdroj), při záporné vektory směřují dovnitř (tzv. dřez).

Je-li $G \subset \mathbb{R}^n$ otevřená množina a $\overrightarrow{\mathbf{F}} = (F_1, \ldots, F_n) : G \to V(\mathbb{R}^n)$ je vektorové pole třídy C_1 na G, lze divergenci vektorového pole $\overrightarrow{\mathbf{F}}$ definovat obecně v $\mathbb{R}^n$ předpisem uvedeným v [16]:

$$\text{div} \overrightarrow{\mathbf{F}} = \frac{\partial F_1}{\partial x_1} + \frac{\partial F_2}{\partial x_2} + \ldots + \frac{\partial F_n}{\partial x_n}. \tag{1.4}$$

Laplaceův operátor

Pomocí Laplaceova operátoru $\triangle$ lze počítat gradient založený na konvoluci dle [18]. Pro výpočet ze čtyřokolí pixelu ve směru kolmém na souřadnicové osy má jeho konvoluční jádro tvar:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{1.5}$$

Hodnota, kterou Laplaceův operátor vypočítá, se rovná aproximaci druhé mocniny gradientu, čili $\triangle f = \nabla^2 f$. Gradient se odhaduje jako součet diferencí ve dvou na sebe kolmých směrech:

$$\triangle f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}. \tag{1.6}$$

1.3 Ostatní pojmy

Isofota

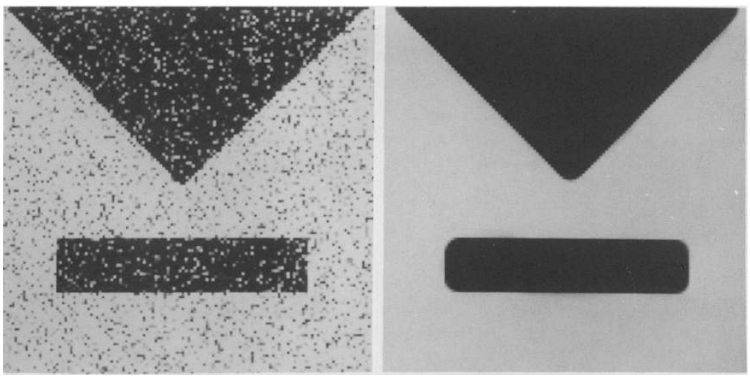
Isofota udává směr vektoru nejmenší prostorové změny [4]. Jestliže v bodě (i,j) diskrétního obrazu $I(i,j)$ udává gradient $\nabla I(i,j)$ směr největší prostorové změny, směrový vektor isofoty $\nabla^{\perp} I(i,j)$ bude vůči němu otočen o 90°. Na směru rotace nezávisí, jelikož se stále jedná o minimální prostorovou změnu.

Patch

Jedná se o pole pixelů o velikosti $m \times m$. Obvykle by velikost takového shluku pixelů měla být lehce vyšší, než největší rozeznatelný element textury neboli texel[7]. Úpravou velikosti patche lze dosáhnout rozdílných výsledků a například autorům metody [11] se dobře osvědčila proměnná velikost tohoto bloku. Výraz patch se bude v následující kapitole vyskytovat především ve spojitosti s metodami založenými na kopírování bloků[7][13].

Anizotropní difúze

Jak bylo popsáno v [2], anizotropní difúze slouží k vyhlazení obrazu, respektive k odstranění šumu. Dále však na rozdíl od izotropické difúze zachovává hrany. Příklad anizotropické difúze je zobrazen na obrázku 1.1.



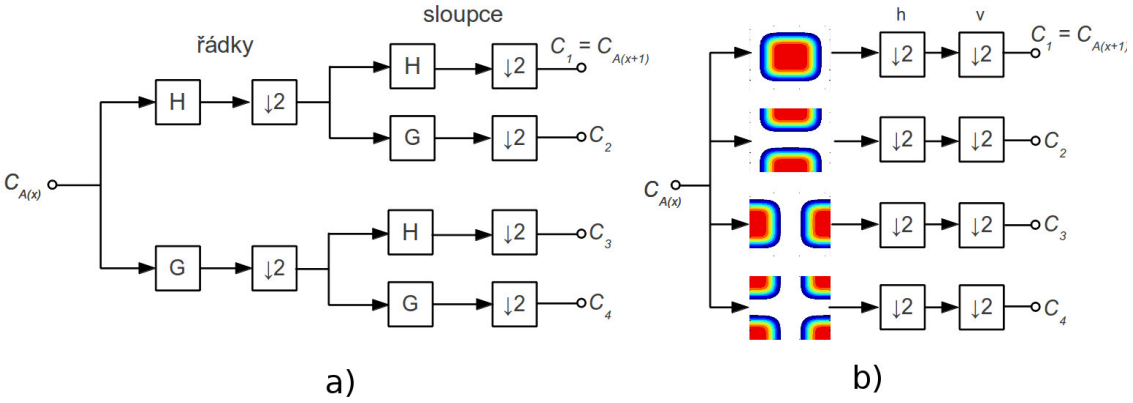
Obr. 1.1: Vpravo zobrazen příklad aplikace anizotropní difúze se selektivním směrem na originální obrázek vlevo. Obrázek použit z původního článku [2].

Wavelet transformace

V [12] se píše, že waveletová nebo také česky vlnková transformace pracuje na principu rozkladu signálu na součet jednotlivých koeficientů - vlnek. Každý waveletový koeficient signálu zastupuje jednu variantu měřítka a posunutí základního waveletu a informuje, kolik se této informace nachází v signálu. Vlnková analýza se provede aplikací tzv. mateřské vlnky. Těch existuje několik druhů, přičemž pro potřeby této práce se bude uvažovat Haarova vlnka detailně popsaná v [14].

Diskrétní waveletová transformace (DTWT) úzce souvisí s bankou filtrů, tedy s lineární filtrací signálu. Jaká banka filtrů se zvolí, závisí na účelu použití transformace. V bance filtrů se nacházejí dva filtry, plnící úlohu dolní a horní propusti.

Provedení vlnkové transformace, popsané v [12], je zobrazeno na obrázku 1.2. Nejdříve je po řádcích provedena diskrétní konvoluce vstupního obrazu a jednotlivých filtrů z banky Wavelet transformace. Tím vzniknou dvě pole koeficientů, která jsou následně po řádcích podvzorkována dvěma. Ten samý proces se opakuje s oběma poli koeficientů po sloupcích. Výsledkem jedné úrovně vlnkové transformace jsou čtyři pole koeficientů. Jejich počet se shoduje s počet bodů vstupního obrazu transformace, případně dolnopropustní větve předchozí úrovně transformace.



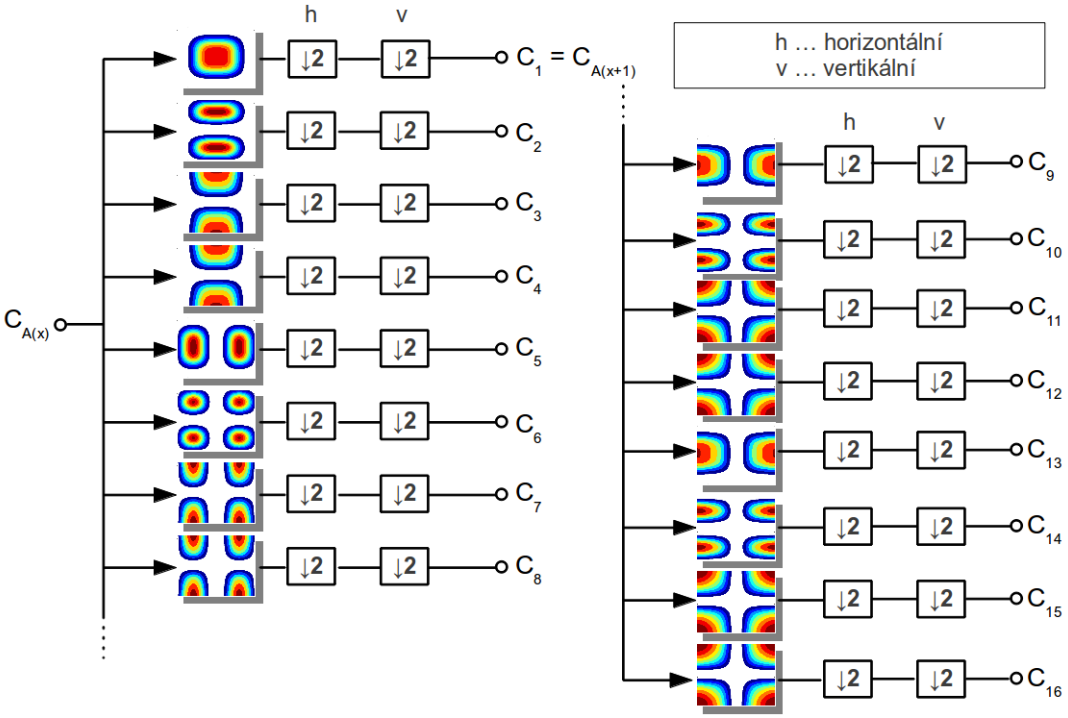
Obr. 1.2: Vlevo je zobrazení průběhu jedné úrovně Wavelet transformace. Vpravo jsou naznačeny modulové charakteristiky filtrů banky „db4“ Wavelet transformace.

Framelet transformace

V článku [1] je napsáno, že se v bance filtrů Framelet transformace nachází čtyři základní jednorozměrné filtry. Jedná se o jednu dolní propust, jednu pásmovou propust a dvě horní propusti. V praktické části této práce byly použity tyto filtry:

$$\begin{aligned}
\text{h0} &= \tfrac{\sqrt{2}}{32}\,[-1, 0, 9, 16, 9, 0, -1],\\
\text{h1} &= \tfrac{\sqrt{2}}{32}\,[-1, -2, 7, 0, -7, 2, 1],\\
\text{h2} &= [\,0.0104677975,\ 0.0370823430,\ 0.0651503417,\ 0.0897493772,\\
&\qquad -0.575618139,\ 0.3731682798, 0\,],\\
\text{h3} &= [\,0,\ 0.0104677975,\ 0.0370823430,\ 0.0651503417,\ 0.0897493772,\\
&\qquad -0.575618139,\ 0.3731682798\,].
\end{aligned}$$

Na obrázku 1.3 jsou zobrazeny naznačené modulové charakteristiky šestnácti filtrů, které vzniknou 2-D konvolucí dvou ze základních filtrů. Některé modulové charakteristiky jsou sice stejné, a přece se liší fázovou charakteristikou. Je to způsobeno horníma propustma h2 a h3, které se liší jen posunem o jeden vzorek. Další úroveň rozkladu lze provést s výsledkem první větve transformace, která na obraz aplikuje v horizontálním i vertikálním směru filtr dolní propust h0.



Obr. 1.3: Naznačené modulové charakteristiky filtrů banky Framelet transformace.

PSNR

Špičkový poměr signálu k šumu (peak signal-to-noise ratio) je poměrem mezi maximální možnou energií obrazu a průměrnou energií šumu v obraze. PSNR se dá vypočítat podle zjednodušeného vztahu, uvedeného v článku [6]:

$$\mathrm{PSNR} = 20 \log_{10} \frac{255}{\sqrt{\mathrm{MSE}}} \qquad [\mathrm{dB}], \tag{1.7}$$

kde číslo 255 označuje nejvyšší možnou hodnotu pixelu a MSE znamená střední kvadratickou chybu (mean squared error), která je v tomto případě počítána jako průměrná hodnota rozdílu dvou obrazů. V případě tří barevných složek obrazu jsou rozdíly obrazu všech těchto složek nejdříve sečteny a následně poděleny třemi.

2 METODY PRO DOPLNĚNÍ OBRAZU

Podle způsobu práce algoritmu jednotlivých metod doplňování obrazu lze tyto metody rozdělit dle [14] do několika základních tříd. Metody popisované v dalších částech této práce budou založené na některých následujících mechanismech.

1. *Parciální difúzní rovnice (PDE)/Variační přístup.* Jedná se z hlediska vývoje o nejstarší přístupy. Na parciálních difúzních rovnicích je například založena metoda Image Inpainting [4]. Použitím difúze se odstraní šum, má to však za následek neexistenci textury v doplňované oblasti. To se znatelně projeví při rekonstrukci větších chybějících oblastí, zejména oblastí s texturou.

2. *Syntéza textury.* Na syntéze textury je například založena metoda Fragmet-Based Image Completion[8]. Tvorba požadované textury v doplňované oblasti probíhá složitým hledáním správných shluků pixelů ve známé oblasti Φ. Jedná se o výpočetně nejnáročnější postup a samotný přístup si neporadí se strukturami v obraze.

3. *Kopírování bloků.* Řešením obou problémů, jak výpočetní náročnosti přístupu syntézy textury, tak práce se strukturami vznikl přístup kopírování bloků. Tento na patchi založený přístup je díky svým výhodám použit v několika metodách jako například Exemplar-Based Image Inpainting [7], Gradient-Based Image Completion by Solving Poisson Equation [13] či Block-Based Image Inpainting in the Wavelet Domain [11]. Cílem algoritmů je najít správný patch požadované velikosti a zkopírovat jej na příslušné místo na okraji hranice $\partial\Omega$ v rekonstruované oblasti.

4. *Regularizace řídkosti (Sparsity regularization).* Jedná se o přímé rozšíření algoritmu MCA (Morphological Component Analysis) za účelem získání chybějících částí obrazu[14].

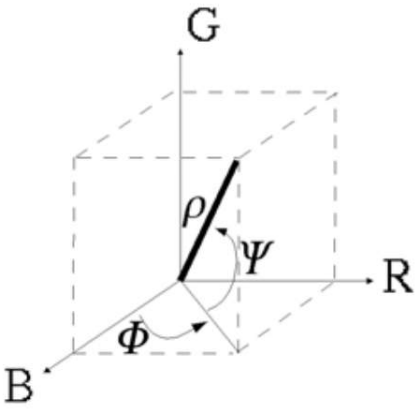
Největším problémem obvykle zůstává správně obnovit tvar objektů. Proto se objevují i způsoby jak tento problém obejít, například metoda Image Completion with Structure Propagation[15] si vyžádá od uživatele načrtnutí hran objektů křižujících neznámou oblast Ω. Výsledkem této interakce uživatele a algoritmu je velmi kvalitní doplnění chybějících oblastí obrazu. Obdobně funguje i nástroj PatchMatch [3].

V následujících sekcích budou popsány algoritmy a komentovány vlastnosti vybraných nejznámějších metod doplňování chybějících částí obrazu.

2.1 Image Inpainting

Metoda Image Inpainting se snaží napodobit základní techniky používané profesionálními restaurátory. K popsání algoritmu se čerpalo z článku [4].

Algoritmus níže popsaného inpaintingu je navržen pro práci pouze s jednou barevnou složkou obrazu. Jestliže se jedná o barevný obraz, obsahující tři barevné složky, musí se provést inpainting pro každou barevnou informaci zvlášť. Aby však nedocházelo k tvorbě nežádoucích barev, bude namísto klasického RGB modelu použit barevný model velmi podobný LUV (obrázek 2.1) s jednou jasovou složkou a dvěma chrominančními složkami.



Obr. 2.1: Vztah mezi barevným modelem RGB a modelem zde použitým. Složky představují jas ρ a chrominance $\sin\Phi$ a $\sin\psi$. Převzato z původního článku [4].

Popis algoritmu

Jedinými vstupy algoritmu [4] jsou obraz I, jež má být opraven a maska Ω, vymezující oblasti doplnění obrazu.

1. Nejdříve je na celý obraz aplikována anisotropní difúze, jejíž účelem je minimalizovat vliv šumu, a tím usnadnit odhad i výběr směru isofot směřující z hranice $\partial\Omega$ do oblasti Ω. Výsledek anisotropní difúze je pro další zpracování uložen zvlášť, tzn. nebudou pozměněny původní oblasti snímku. Následně se spustí výpočet, který iterativně upravuje oblast Ω.

2. Následuje výpočet pro každý obrazový bod $I^n(x, y)$ uvnitř oblasti Ω. Zde n značí čas, neboli číslo předchozí iterace. Výpočet následující iterace je vždy dán rovnicí:

$$I^{n+1}(x,y) = I^n(x,y) + \Delta t I_t^n(x,y), \qquad \forall [x,y] \in \Omega, \tag{2.1}$$

kde Δt je míra zlepšení a $I_t^n(x,y)$ znamená aktualizace obrazu $I^n(x,y)$. S každým zvýšením n je získán lepší obraz. Aktualizace $I_t^n(x,y)$ ve zjednodušené formě je dána vztahem:

$$I_t^n(x,y) = \overrightarrow{\delta L^n}(x,y) \cdot \overrightarrow{N}^n(x,y), \tag{2.2}$$

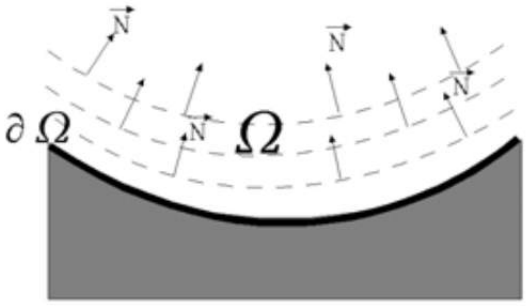
zde $\overrightarrow{\delta L^n}(x,y)$ vyjadřuje rozsah změny informace $L^n(x,y)$, kterou je třeba rozšířit a $\overrightarrow{N}^n(x,y)$ je směr rozšíření informace. Za účelem hladkého rozšíření informace vykonává výraz $L^n(x,y)$ funkci odhadu hladkosti. Proto autoři algoritmu použili jednoduchou diskrétní implementaci Laplaceova operátoru:

$$L^n(x,y) := I_{xx}^n(x,y) + I_{yy}^n(x,y), \tag{2.3}$$

kde dolní indexy značí derivace. Následuje výpočet změny $\overrightarrow{\delta L^n}(x,y)$ v závislosti na $\overrightarrow{N}$. Pro zjištění směru $\overrightarrow{N}$ tento algoritmus využívá časově proměnlivý výpočet isofoty. Gradientní vektor $\nabla I^n(x,y)$ udává směr největší prostorové změny, zatímco vektor na něj kolmý $\nabla^\perp I^n(x,y)$ vyjadřuje směr nejmenší prostorové změny, tedy směr isofoty. Časově proměnné pole $\overrightarrow{N}$ je vyjádřeno vztahem

$$\overrightarrow{N}(x,y,n) = \nabla^\perp I^n(x,y). \tag{2.4}$$

Nesprávným způsobem volby směru $\overrightarrow{N}$ by bylo použití normály tečny určené vzdálenosti od hranice $\partial\Omega$, jak je zobrazeno na obrázku 2.2. Výsledek takového přístupu lze vidět na obrázku 2.3,



Obr. 2.2: Příklad nesprávného určení směru rozšíření informace do Ω. Směr dán normálou tečny určené vzdálenosti od hranice $\partial\Omega$. Převzato z původního článku [4].

Obr. 2.3: Důsledek nevhodné volby směru rozšíření informace. Vlevo bíle vyznačena oblast Ω; vpravo doplněná oblast nerespektuje struktury obrazu. Obrázek převzat z původního článku [4].

3. Po každých několika iteracích je znovu provedena anisotropní difúze v oblasti Ω, tedy je aplikována upravená rovnice anisotropní difúze:

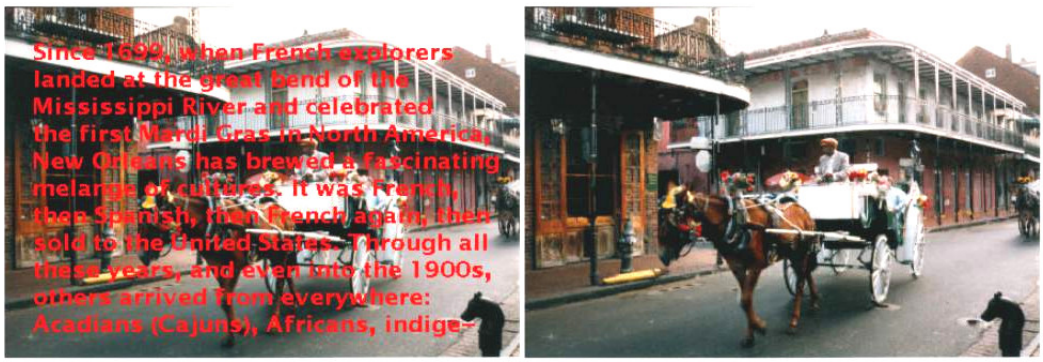
$$\frac{\partial I}{\partial t}(x,y,t) = g_\epsilon(x,y)\kappa(x,y,t) \mid \nabla I(x,y,t) \mid, \qquad \forall (x,y) \in \Omega^\epsilon, \tag{2.5}$$

kde Ω^ϵ znamená dilataci (rozšíření) do Ω v kruhu o poloměru ϵ a funkce κ vyjadřuje rovinné (euklidovské) zakřivení isofot vstupního obrazu I. Posledním prvkem rovnice je $g_\epsilon(x,y)$, tedy vyhlazovací funkce pro oblast Ω^ϵ. Jejím účelem je do rovnice vložit Dirichletovu okrajovou podmínku. Tudíž se funkce $g_\epsilon(x,y) = 0$ pro $\partial\Omega^\epsilon$ a zároveň $g_\epsilon(x,y) = 1$ v Ω.

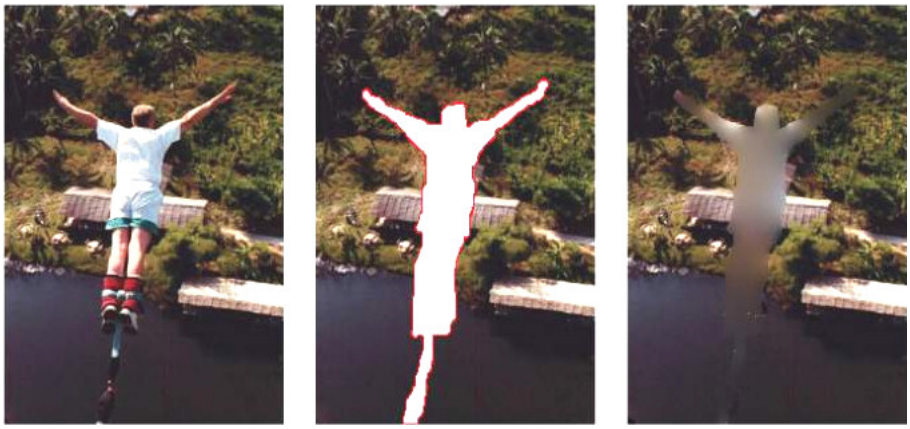
Vyhodnocení metody

Image Inpaintig jako jedna z průkopnických metod správně funguje v případě potřeby opravy škrábanců či úzkých oblastí poškozeného obrazu. Na obrázku 2.4 autoři úspěšně demonstrovali použití popsané metody. Při detailním prozkoumání výsledného obrazu je však zřejmé, že má metoda slabiny v případě opravy ostrých hran některých objektů a nedokáže nahradit texturu. Na druhou stranu dokáže dobře opravit neostré obrázky o nižším rozlišení, jak ukazuje obrázek 2.5. Opravu rozsáhlé oblasti znázorňuje obrázek 2.6. Je jasné, že pro tyto účely se metoda inpaintingu nehodí.

Obr. 2.4: Obnovení úzkých oblastí v obraze pomocí metody Image Inpainting. Obrázek převzat z původního článku [4].



Obr. 2.5: Příklad odstranění písma v obraze pomocí metody Image Inpainting. Obrázek převzat z původního článku [4].



Obr. 2.6: Neuspokojivé doplnění rozsáhlé oblasti obrazu se složitou strukturou pomocí metody Image Inpainting. Obrázek převzat z článku [13].

2.2 Fragment-Based Image Completion

K doplňování větších částí obrazu byla následně navržena autory Drori a spol. [8] metoda Fragment-Based Image Completion. Původní myšlenkou autorů byl algoritmus schopný odstranit nežádoucí objekty v popředí či oblasti v pozadí obrazu. Narozdíl od předchozí metody Image Inpainting [4] se tato snaží syntetizovat texturu. Cílem metody je na základě znalosti celého obrazu doplnit neznámé oblasti. Hodnoty získané aplikováním inverzní masky $\overline{\alpha}$ jsou použity k vypočtení mapy důvěryhodnosti β. Nejdříve aproximuje neznámou oblast obrazu a poté se snaží ji zaplnit vhodnými fragmenty obrazu. Při popisu algoritmu bylo čerpáno z původního článku [8].

Popis algoritmu

1. Prvním krokem je aproximace. Rychlý odhad barev v chybějící oblasti funguje na základě prostého iterativního filtračního procesu založeného na známých hodnotách. Obsah takto vyplněné oblasti nám dává možnost hledat v okolí fragmenty obrazu s podobnou hodnotou.

2. Definujeme mapu důvěryhodnosti β přidělením hodnoty každému pixelu i:

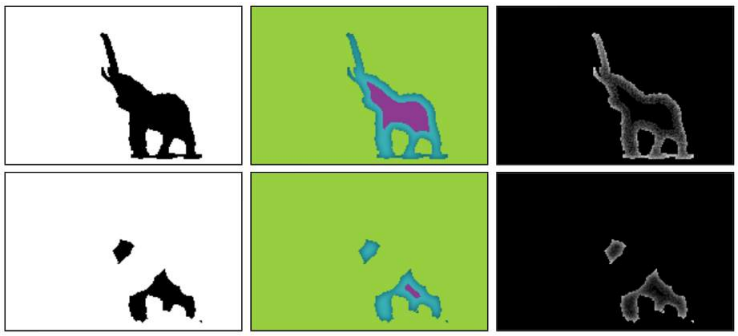
$$\beta_i = \begin{cases} 1 & \text{jestliže } \overline{\alpha}_i = 1 \\ \Sigma_{j \in N(i)} g_j \overline{\alpha}_j^2 & \text{jinak,} \end{cases} \tag{2.6}$$

   kde N je okolí kolem pixelu i a g je exponenciální stranový pokles Gaussova filtru. Mapa určuje, kolik důvěryhodnosti vložit do obrazové informace každého pixelu a je použita k porovnávání a výběr správných fragmentů.

3. V dalším kroku je z mapy důvěryhodnosti β vypočtena mapa kandidátů (překlad použit z [9]) následujícím způsobem:

$$v_i = \begin{cases} 0 & \text{jestliže } \beta_i > \mu(\beta) \\ \beta_i + \rho[0, \sigma(\beta)] & \text{jinak.} \end{cases} \tag{2.7}$$

   V místech, kde je hodnota důvěryhodnosti β_i vyšší, než průměrná hodnota $\mu(\beta)$ funkce β, je nastavena hodnota mapy kandidátů v_i na 0. Jinak je určena hodnota v_i, přidáním náhodného rovnoměrného šumu v rozmezí 0 a směrodatné odchylky $\sigma(\beta)$, k hodnotě důvěryhodnosti β_i.
   Obrázek 2.7 zobrazuje ve dvou krocích inverzní masku $\overline{\alpha}$, mapu funkce β v logaritmickém měřítku a také mapu kandidátů funkce v.



Obr. 2.7: Zleva: Inverzní maska, mapa důvěryhodnosti v logaritmickém měřítku a mapa kandidátů. Převzato z původního článku [8].

4. Následuje hledání, kdy se porovnává důvěryhodnost každého pixelu. Nyní se pro každý cílový fragment T hledá nejlepší shoda se zdrojem S, o parametrech $r = (l, x, y, \Theta)$. Odtud x, y znázorňují všechny pozice, dále je zde pět měřítek l a osm orientací Θ. Mapa důvěryhodnosti je zde použita pro porovnání páru fragmentů s ohledem na příslušné pixely $s = S(i)$ a $t = T(i)$ v jejich okolí N. Pro každý takový pár pixelů existují β_s a β_t, značící jejich důvěryhodnost a funkce $d(s, t)$ určující jejich podobnost, respektive podobnost jejich rysů. Těmito rysy jsou barva, jas a gradienty (první derivace v horizontálním, vertikálním a diagonálním směru). Pozice, velikost a orientace cílového fragmentu je nalezena pomocí funkce:

$$r^* = \arg\min_r \sum_{s=S_r(i), t=T(i), i \in N} (d(s,t)\beta_s\beta_t + (\beta_t - \beta_s)\beta_t). \tag{2.8}$$

   Ve zmíněné funkci použitý výraz $d(s, t)\beta_s\beta_t$ penalizuje rozdílné hodnoty odpovídajících si pixelů jak v cílovém, tak zdrojovém fragmentu. Výraz $(\beta_t - \beta_s)\beta_t$ odměňuje pixely s vyšší důvěryhodností ve zdrojovém, než v cílovém fragmentu a penalizuje opak.

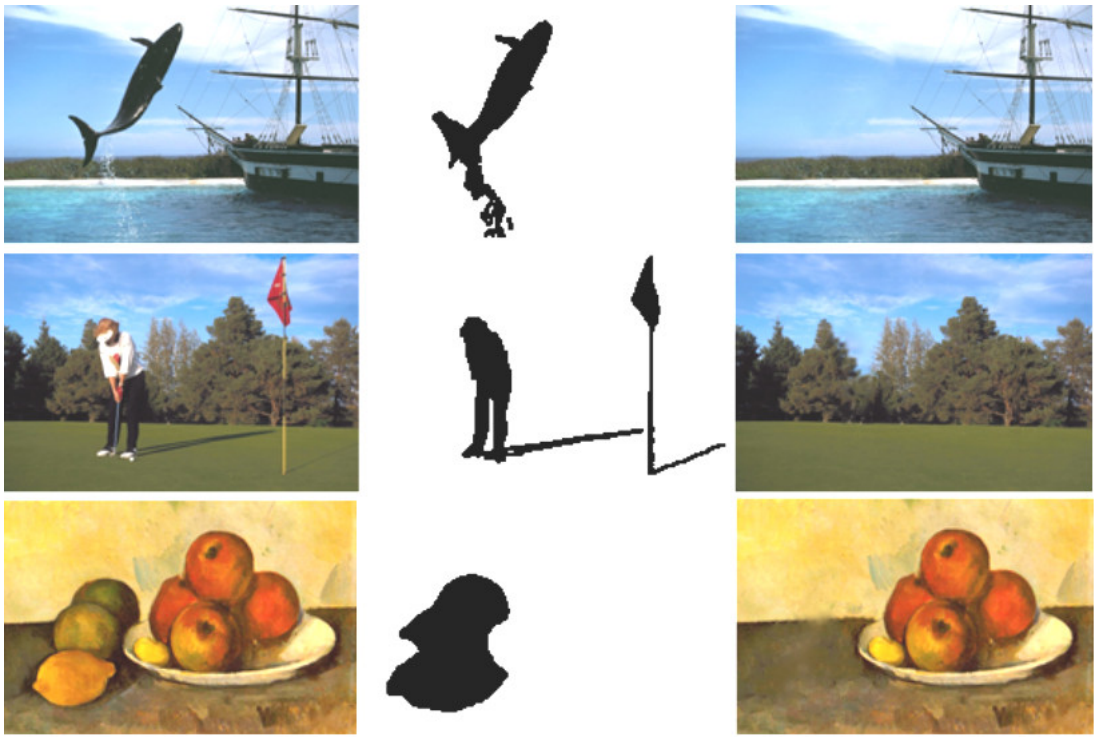
5. V posledním kroku dochází ke složení fragmentů. Jinými slovy, jsou nalezené cílové fragmenty T uloženy na místo zdrojových fragmentů S. Za účelem jemného splynutí fragmentů je zde použita Laplaceova pyramida.

Vyhodnocení metody

Fragment-Based Image Completion dosahuje docela kvalitní opravy větší plochy chybějící textury (příklady zobrazeny na obrázku 2.8), lze ji však rozumně aplikovat

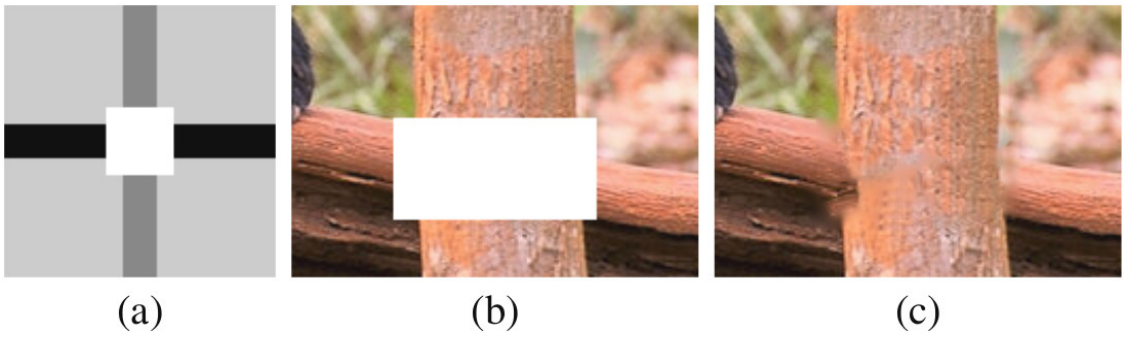
jen na obrázky malého rozlišení. Použitý algoritmus je dle výsledků [8] či [13] velmi náročný na výpočet a s rostoucím rozlišením roste výpočetní čas kvadraticky.

Tato metoda se nehodí pro doplnění chybějících částí objektů v prostoru, jak je znázorněno na obrázku 2.9, nesvědčí jí však ani oblasti, kde se vyskytují nejednoznačné struktury viz obrázek 2.10.



Obr. 2.8: Příklad doplnění některých známých obrázků pomocí metody Fragment-Based Image Completion. Zleva: původní obrázek, maska a výsledek. Obrázky z původního článku [8].

Obr. 2.9: Příklad neúspěšného doplnění objektu v obraze. Obrázek převzat z původního článku [8].

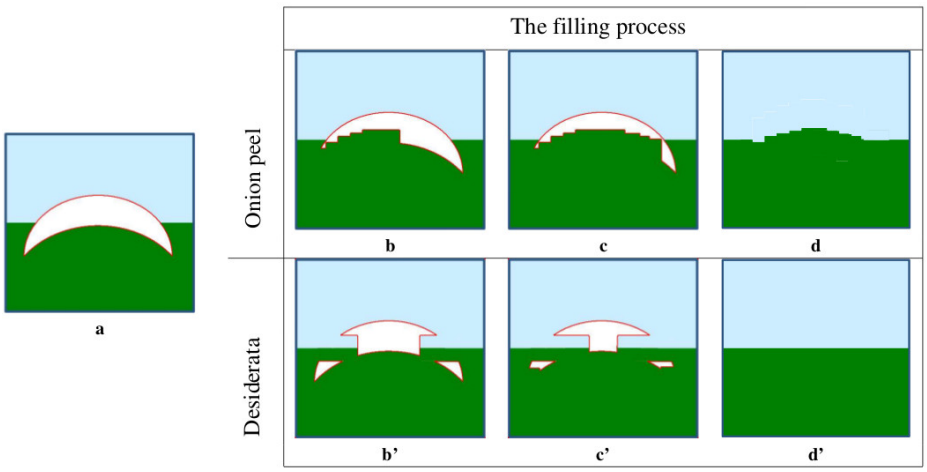


Obr. 2.10: Neuspokojivé doplnění nejednoznačné struktury pomocí metody Fragment-Based Image Completion. Vlevo je zobrazen druh problému, uprostřed je označena neznámá oblast a vpravo je doplněna. Převzato z původního článku [8].

2.3 Exemplar-Based Image Inpainting

Metoda popsaná v [7] kombinuje výhody dvou používaných přístupů doplňování obrazu, inpainting pro doplnění malých oblastí a syntézu textury, jež dokáže doplňovat rozsáhlejší díry v obraze. Exemplar-Based Image Inpainting využívá rozšiřování struktury a textury, přičemž velmi záleží na pořadí, ve kterém dochází k plnění.

Starší algoritmy, spoléhající se na rozšiřování textury, využívaly plnění po soustředných vrstvách, obvykle nazývané jako slupky cibule. Algoritmus aplikovaný na určité vypouklé oblasti přechodů struktur však způsobuje nežádoucí deformaci zobrazenou na obrázku 2.11.



Obr. 2.11: Zobrazení nutnosti správného pořadí plnění v případě vypouklých cílových oblastí. Obrázek převzat z původního článku [7].

Popis algoritmu

Vstupem algoritmu jsou obraz I a cílová oblast Ω. Jednotlivé patche Ψ budou vybírány ze zdrojové oblasti $\Phi = I - \Omega$. Velikost takovýchto záplat je nastavena pevně. Autoři této metody[7] doporučují zvolit velikost nepatrně větší, než je největší rozeznatelný element textury, obvykle nazývaný texel. Při psaní popisu algoritmu níže se bude vycházet z původního článku [7].

Dokud průběh algoritmu neskončí, probíhají v iteraci následující tři kroky:

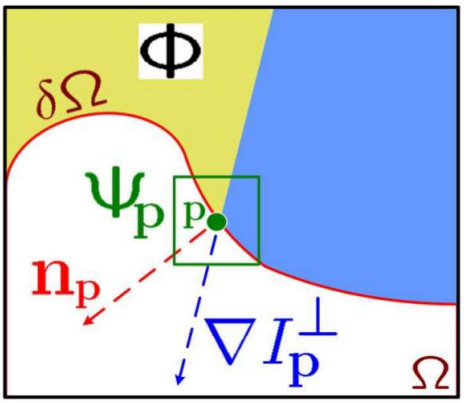
1. **Výpočet priority patche.** Na obrázku 2.12 se nalézá patch Ψ_p, umístěn v hraničním bodě p, $\mathrm{p} \in \partial\Omega$. Priorita P patche bude vypočítána následujícím způsobem:

$$P(\mathrm{p}) = C(\mathrm{p})D(\mathrm{p}). \tag{2.9}$$

   Výraz $C(\mathrm{p})$ vyjadřuje důvěryhodnost a $D(\mathrm{p})$ data. Tyto výrazy jsou definovány následujícím způsobem:

$$C(\mathrm{p}) = \frac{\Sigma_{\mathrm{q}\in\Psi_\mathrm{p}\cap(I-\Omega)}C(\mathrm{q})}{\mid \Psi_\mathrm{p} \mid}, \qquad D(\mathrm{p}) = \frac{\mid \nabla I_\mathrm{p}^{\perp} \cdot \boldsymbol{n}_\mathrm{p} \mid}{\alpha}. \tag{2.10}$$

   Zde $\mid \Psi_\mathrm{p} \mid$ znamená plochu Ψ_p, $\boldsymbol{n}_\mathrm{p}$ je vektor kolmý na hranici $\partial\Omega$ v bodě p. Priorita $P(\mathrm{p})$ je spočtena pro každý hraniční patch, respektive pro každý hraniční bod p je vypočten vlastní patch.



Obr. 2.12: Diagram vysvětluje výše zavedené pojmy. Obrázek převzat z původního článku [7].

2. **Rozšíření textury a struktury.** Nalezený patch Ψ_p s největší prioritou se naplní informací vybranou ze zdrojové oblasti Φ. Rozšíření se děje přímým

kopírováním informace ze zdrojové oblasti. Zdrojový patch $\Psi_{\hat{q}}$ musí být kompletně obsažen ve zdrojové oblasti Φ a hledá se následujícím způsobem:

$$\Psi_{\hat{q}} = \arg\min_{\Psi_q \in \Phi} d(\Psi_{\hat{p}}, \Psi_q). \tag{2.11}$$

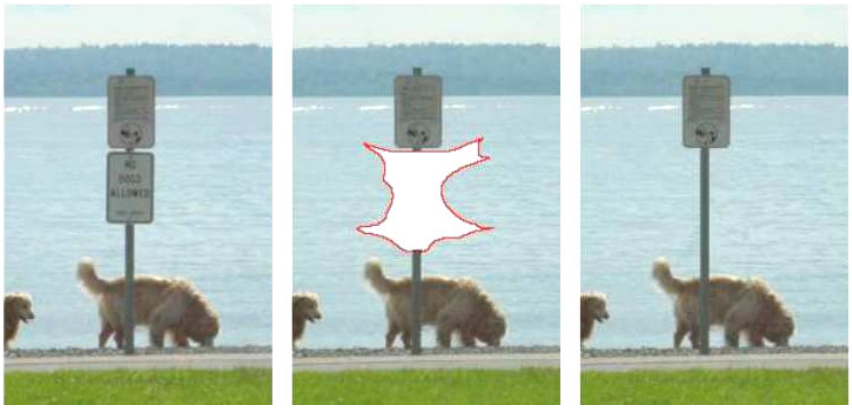
kde je vzdálenost $d(\Psi_a, \Psi_b)$ mezi dvěma záplatami Ψ_a a Ψ_b definována jako součet čtverců rozdílů (SSD).

3. **Update hodnot důvěryhodnosti.** Po naplnění cílového patche $\Psi_{\hat{p}}$ je důvěryhodnost $C(p)$ v této oblasti upravena následujícím způsobem:

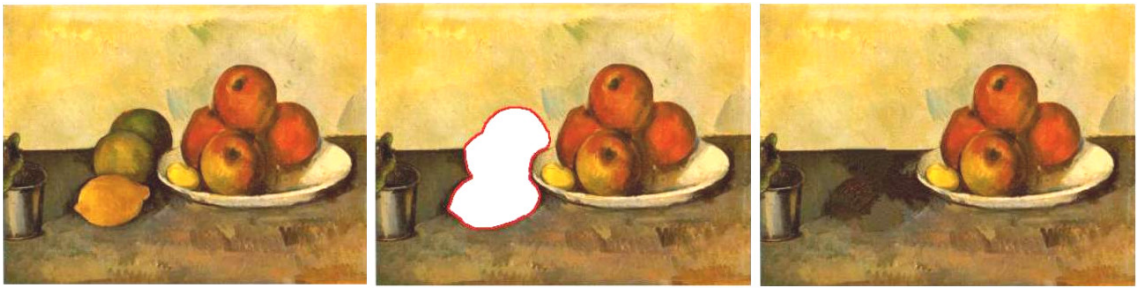
$$C(p) = C(\hat{p}) \quad \forall p \in \Psi_{\hat{p}} \cap \Omega. \tag{2.12}$$

Vyhodnocení metody

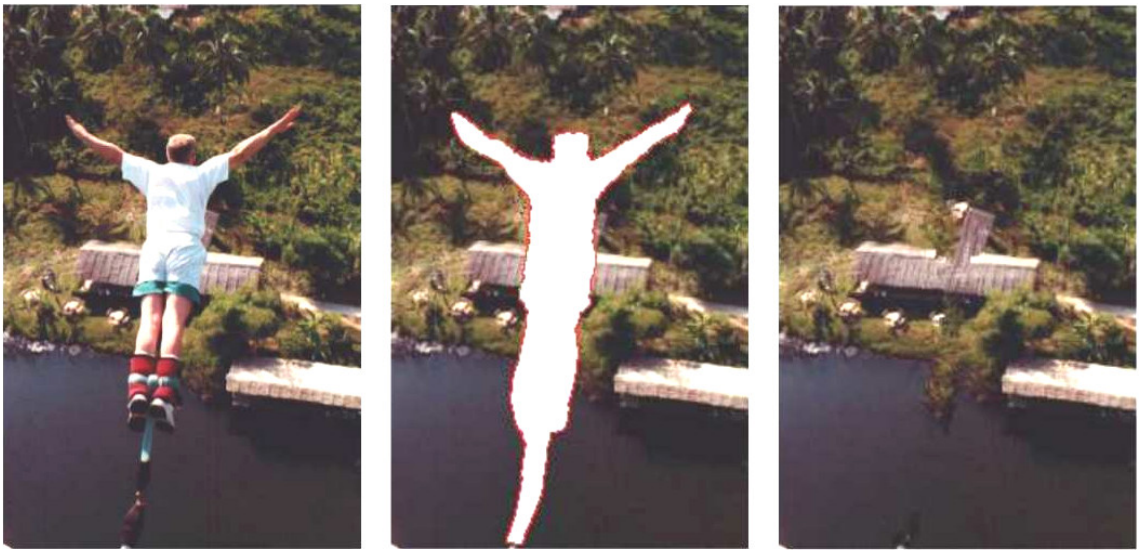
Předností výše popsané metody Exemplar-Based Image Inpainting je nižší náročnost na výpočetní výkon (jak bylo popsáno v [13]) a v mnohých případech správné doplnění struktury i textury chybějící části obrazu, jak je demonstrováno na obrázku 2.13. Na obrázku 2.14 lze vidět, že si tato metoda dokáže dobře poradit s doplněním textury, správně je také vytvořena horní hrana. Celkově je obraz v této oblasti ostrý, jelikož v žádném kroku algoritmu není aplikováno vyhlazení narozdíl od předchozích metod. V porovnání s Fragment-Based Image Completion, dosáhla v tomto případě lepšího výsledku ve znatelně kratším čase. Exemplar-Based Image Inpainting však stále nedokáže uspokojivě opravit některé složité struktury, jak je ukázáno na obrázku 2.15, kde vznikají artefakty a tvar doplněné oblasti nezapadá logicky do celkového obrazu.



Obr. 2.13: Příklad správného doplnění jednodušší struktury pomocí metody Exemplar-Based Image Inpainting. Obrázek převzat z původního článku [7].



Obr. 2.14: Příklad úspěšného doplnění hrany a vyplnění textury metodou Exemplar-Based Image Inpainting. Obrázek převzat z původního článku [7].



Obr. 2.15: Nahrazení větší plochy obrazu s náročnou strukturou pomocí metody Exemplar-Based Image Inpainting. Obrázek převzat z původního článku [7].

2.4 Gradient-Based Image Completion by Solving Poisson Equation

V roce 2007 byla čínskými autory[13] uvedena metoda doplňování obrazu založená na gradientech řešením Poissonovy rovnice. Přičemž celý smysl algoritmu spočívá ve dvou hlavních fázích: Nejdříve jsou vytvořeny gradientní mapy v oblasti Ω, naplněním oblasti informací pomocí patch-based algoritmu. V druhé části je obraz rekonstruován z gradientních map řešením Poissonových rovnic. Nově navržené porovnávací kritérium pracuje jak s gradienty, tak s barevnou informací. To má za následek lepší výsledky doplňování obrazu. Při popisu algoritmu bylo čerpáno výhradně z původního článku [13].

Popis algoritmu

Stejně jako v předchozích případech je vstupem algoritmu obraz I a maska Ω. Známá oblast nese označení Φ a hranice se značí $\partial\Omega$. Výstupem algoritmu je obraz I' s vyplněnou oblastí Ω. V následujících krocích se bude operovat se zdrojovým patchem Ψ_s a cílovým patchem Ψ_t. Výraz $G(...)$ pro zjednodušení zápisu bude vyjadřovat informace a operace v oblasti gradientů.

Celý algoritmus lze rozdělit do následujících bodů:

1. **Výběr cílového patche** Ψ_t. Takový patch, se středem ležícím na hranici $\partial\Omega$, se nejprve vybere podle nejvyšší priority plnění $P(\mathrm{p}_c)$. Parametr důvěryhodnosti $C(\mathrm{p})$ v bodě p se určí následujícím způsobem:

$$C(\mathrm{p}) = \begin{cases} 0, & \forall \mathrm{p} \in \Omega \\ 1, & \forall \mathrm{p} \in I \setminus \Omega \end{cases} \tag{2.13}$$

   Druhým potřebným parametrem definovaným pro bod p je výraz gradientu $G(\mathrm{p})$. Gradientní výraz $G(\mathrm{p})$ se definuje následujícím způsobem:

$$G(\mathrm{p}) = \frac{1}{\mid A \mid} \sum_{\mathrm{q} \in A} \sqrt{G_x^2(\mathrm{q}) + G_y^2(\mathrm{q})}. \tag{2.14}$$

   Zde A vyjadřuje sousední oblast okolo bodu p a má velikost shodnou s patchem Ψ_t, respektive Ψ_s. Výraz $\boldsymbol{G} = [G_x, G_y]$ definuje gradientní pole obrazu v horizontálním i vertikálním směru, za použití prosté dopředné diference.
   Plnící priorita $P(\mathrm{p}_c)$ pro bod $\mathrm{p}_c \in \partial\Omega$ bude získána z následujících rovnic:

$$P(\mathrm{p}_c) = \Upsilon(\mathrm{p}_c) \cdot G(\mathrm{p}_c), \tag{2.15}$$

$$\Upsilon(\mathrm{p}_c) = \frac{\Sigma_{\mathrm{q} \in \Psi_t \cap (I \setminus \Omega)}\, C(\mathrm{q})}{\mid \Psi_t \mid}. \tag{2.16}$$

   Zde výraz gradientu $G(\mathrm{p}_c)$ vyjadřuje rozsah množství hran a strukturální informace v okolí bodu p_c a nakonec $\mid \Psi_t \mid$ dává velikost patche Ψ_t. Účelem výpočtu hodnoty priority patche je určit, která místa je důležité naplnit nejdříve a tím správně spojit linie porušené struktury v obrazu.

2. **Hledání zdrojového patche** Ψ_s. Hledá se největší podobnost zdrojového patche Ψ_s v oblasti Φ a vybraného cílového patche Ψ_t se středem $\mathrm{p}_c \in \partial\Omega$ s nejvyšší prioritou plnění. Autoři této metody vyvinuli pro hledání zdrojového patche měření exponenciální podobnosti, jež zahrnuje jak rozdíly v barvě, tak v gradientech mezi dvěma patchi. Tato podobnost je definována jako

$$s(\Psi_s, \Psi_t) = \exp^{d_c(\Psi_s,\Psi_t)+d_g(\Psi_s,\Psi_t)}, \tag{2.17}$$

kde prvky v exponentu jsou vyjádřeny

$$d_c(\Psi_s, \Psi_t) = \sum_{(x,y)} \mid\mid \Psi_s^c(x,y) - \Psi_s^c(x,y) \mid\mid, \tag{2.18}$$

$$d_g(\Psi_s, \Psi_t) = \sum_{(x,y)} \mid\mid \Psi_s^g(x,y) - \Psi_s^g(x,y) \mid\mid . \tag{2.19}$$

Zde výrazy Ψ_s^c a Ψ_t^c vyjadřují barevnou informaci zdrojového, respektive cílového patche v RGB. Výrazy Ψ_s^g a Ψ_t^g zase reprezentují gradientní vektory. S ohledem na tato kritéria může být zdrojový patch Ψ_s pro plnění gradientů v Ψ_t vybrán dle vztahu

$$\Psi_s = \arg\min_{\Psi_i \in \Phi} \frac{s(\Psi_i, \Psi_t)}{\mid \Psi_i \mid}. \tag{2.20}$$

Vhodný zdrojový patch Ψ_s se tedy uloží do map gradientů na místo cílového patche Ψ_t a následně se oba kroky algoritmu iterativně opakují, dokud nebudou zaplněny gradientní mapy v celé oblasti Ω.

3. **Rekonstrukce obrazu z gradientů**. Nyní již je kompletní gradientní mapa $\boldsymbol{G'} = [G'_x, G'_y]$, jež je vyžadována pro vytvoření obrazu I' za použití řešení Poissonovy rovnice. Jelikož se pohybujeme v rovině, nemuselo by být modifikované gradientní vektorové pole $\boldsymbol{G'} = [G'_x, G'_y]$ integrovatelné. Z toho důvodu musí být použita některá metoda pro minimalizaci. Autoři algoritmu navrhli použít minimalizaci $\mid\mid \nabla I' - \boldsymbol{G} \mid\mid$ tak, že $\boldsymbol{G} \approx \nabla I'$. Po tomto kroku už může být obraz I' vypočítán užitím Poissonovy diferenciální rovnice

$$\nabla^2 I' = \mathrm{div}([G'_x, G'_y]). \tag{2.21}$$

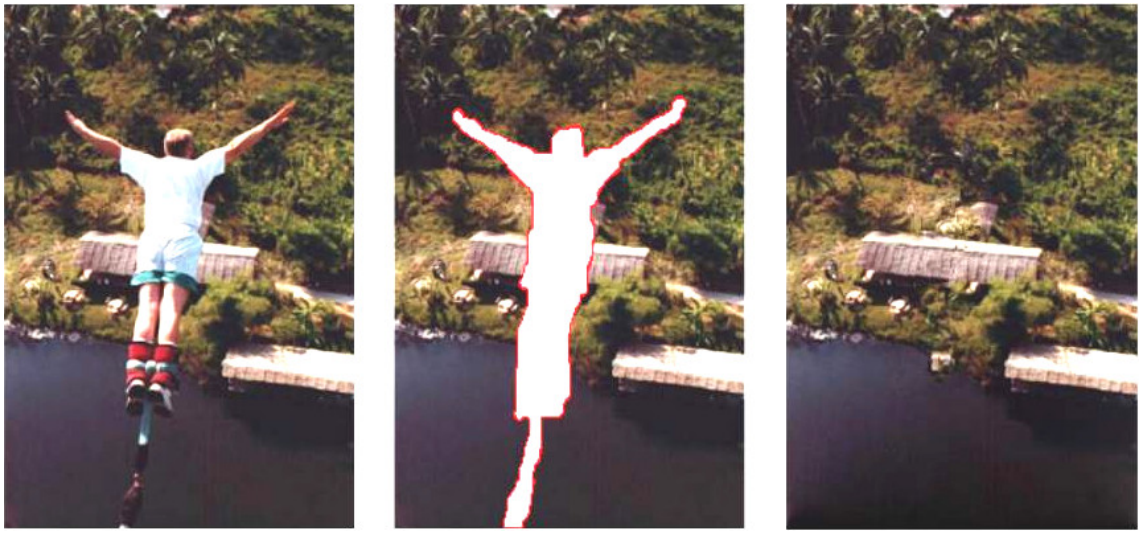
Jak Laplace ∇^2, tak div jsou lineární operátory, jejich aproximací se získá rozsáhlá soustava lineárních rovnic. Laplaceova rovnice je vyřešena použitím full Multigrid metody (FMG). Poissonova rovnice je řešena jen v oblasti Ω.

Vyhodnocení metody

Použitím popsané metody dosáhli její autoři velmi dobrých výsledků jak v nahrazování textury, tak struktury. Dobrým příkladem je obrázek 2.17, který se poprvé

povedlo celkem obstojně zrekonstruovat. Jelikož metoda založená na gradientech využívá stejně jako metoda Exemplar-Based Image Completion [7] přístup kopírování patche, proto sdílí s touto metodou některé vlastnosti. Například je algoritmus velmi rychlý, třebaže je znát mírné zpomalení oproti metodě [7] dané výpočtem Poissonovy rovnice. Naopak pokud se nevyskytuje ve známé části obrazu dostatečné množství vzorků, není možné rekonstruovat vyžadovanou texturu či strukturu a tvoří se artefakty. Dále přístup nezvládá správně rekonstruovat strukturu v místě s různou hloubkou obrazu viz obrázek 2.18 a problémy této metodě dělají i zakřivené struktury jak si lze například všimnout z doplněné horní hrany ovoce na obrázku 2.16.

Obr. 2.16: Příklad úspěšného nahrazení textury. Lze si však všimnout v doplněné části vpravo nedokonalé opravy zakřiveného horního okraje ovoce. Obrázek převzat z původního článku [13].



Obr. 2.17: Úspěšné nahrazení struktury i textury. Při detailním zkoumání si lze povšimnout jemných artefaktů v doplněném obraze vpravo. Obrázek převzat z původního článku [13].

Obr. 2.18: Neuspokojivé doplnění nejednoznačné struktury v místě překřížení objektů s různou hloubkou obrazu. Obrázek použit z původního článku [13].

2.5 Inpainting by Flexible Haar-Wavelet Shrinkage

Do dnes bylo představeno několik algoritmů pracujících s wavelety. Mohou se však lišit svým přístupem k řešení daného problému. Například metoda [11] je založena na kopírování bloků a vlnkovou transformaci využívá k výběru správného patche. V této sekci popisovaná metoda inpaintingu [6] založená na flexibilním smršťováním Haarovy vlnky staví na myšlence anizotropické regularizace a difúze. K vypracování popisu algoritmu posloužil především původní článek [6].

Popis algoritmu

Dvourozměrný obraz **u** definovaný jako $\{0, ..., n\} \times \{0, ..., n\}$ bodů je transformován na vektor $\boldsymbol{u} \in \mathbb{R}^N$ kde $N = n^2$. Jestliže oblast známých dat udává množina $C \subset \{1, ..., N\}$ a $\varepsilon(j)$ znamená šum, potom nekompletní obraz f je dán funkcí

$$f(j) = \begin{cases} u(j) + \varepsilon(j) & \text{jestliže } j \in C, \\ \text{hledaný} & \text{jinak,} \end{cases} \tag{2.22}$$

Pro použití této metody může být podle potřeby použit jeden ze dvou následujících algoritmů. První se použije při čistém vstupním obraze, druhý při zašuměném.

I. Čistý vstupní obraz

1. Inicializace vstupních obrazových dat u_0
2. Pro $r = 0, ...$ probíhá iterace následujících kroků, dokud nekonverguje
   (a) Řešení obnovy aktuálního obrazu u_r pro získání $\widehat{u}_{r+1}$
   (b) Nastavení

$$u_{r+1}(j) = \begin{cases} f(j) & \text{jestliže } j \in C, \\ \widehat{u}_{r+1}(j) & \text{jinak.} \end{cases} \tag{2.23}$$

3. Výstupní obraz u^*

II. Zašuměný vstupní obraz

Pro zašuměná data je použit stejný algoritmus, je však nutné opět aplikovat na výstupní obraz u^* krok 2a. Získá se tak výstup: $u^\diamond = \widehat{u}^*$.

Při vývoji řešení 2a autoři použili část algoritmu metody [5]. Pro další pokračování je nutné si vysvětlit některé prvky. Matice A , $A \in R^{M,N}, M \geq N$, je nazývána operátor analýzy (dekompozice), její adjungovaná matice A^* vyjadřuje operátor syntézy (rekonstrukce)[5]. Každý obraz převedený na vektor $\boldsymbol{u} \in R^N$ lze zapsat jako $\boldsymbol{u} = A^*d$. Také je třeba uvést, že $A^*A = \mathcal{I}$, přičemž je třeba zdůraznit že $AA^* \neq \mathcal{I}$. $\mathcal{I}$ je jednotková matice. Dále existuje diagonální matice Λ obsahující prvky vektoru $\lambda, (\lambda_j)_{j=1}^M$ jako položky v diagonále. Pro získání d následujícího kroku je potřeba řešit rovnici:

$$d_{r+1} = \arg\min_{d \in R^M} \frac{1}{2} \mid\mid c - d \mid\mid_2^2 + \mid\mid \Lambda d \mid\mid_1 . \tag{2.24}$$

Tuto úlohu autoři řeší použitím operátoru měkkého prahování $\mathcal{T}_\Lambda$. Funkce operátoru $\mathcal{T}_\Lambda(c)$ je definována jako:

$$\mathcal{T}_\Lambda(c_j) = \frac{1}{2}((c_j - \lambda_j) + \mid c_j - \lambda_j \mid + (c_j + \lambda_j) - \mid c_j + \lambda_j \mid), \quad j = 1, ..., M. \tag{2.25}$$

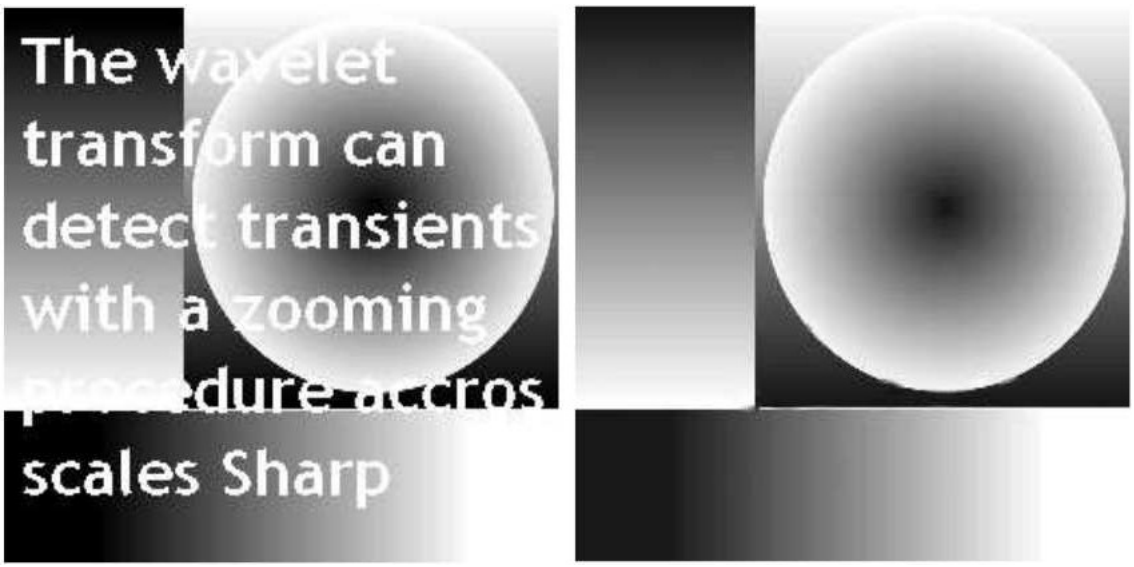
Použitím měkkého prahování se získá vztah pro řešení obnovy obrazu z bodu 2a:

$$\widehat{u}_{r+1} = A^* \mathcal{T}_\Lambda(A u_r) \tag{2.26}$$

Pro shrnutí tedy dochází během jedné iterace k rozkladu obrazu předchozí iterace (u první iterace vstupního obrazu) po sloupcích na vektor, provede se dekompozice získané informace pomocí waveletové transformace. Následuje měkké prahování vytvořených koeficientů a výsledná syntéza.

Vyhodnocení algoritmu

Účelem algoritmu je doplnit úzké plochy obrazu jako písmo nebo šrámy, nehodí se na doplnění rozsáhlejších oblastí. Výhodou použití vlnkové transformace je rychlost výpočtu a kvalitní doplnění úzkých oblastí bez složité textury viz obrázek 2.19. Z obrázku je zřejmé, že algoritmus může mít problémy v některých případech doplnění hrany. V detailu obrázku 2.20 jde vidět uspokojivé doplnění hrany.



Obr. 2.19: Doplnění kulaté hrany pomocí algoritmu Inpainting by Flexible Haar-Wavelet Shrinkage. Obrázek převzat z původního článku [6].

Obr. 2.20: Detail obnovy hran. Na snímku uprostřed použita kubická interpolace, vpravo výsledek algoritmu I. Obrázek použit z původního článku [6].

3 VÝSLEDKY BAKALÁŘSKÉ PRÁCE

Tato kapitola se zabývá praktickou částí bakalářské práce, jejíž výstupem byla implementace vybrané metody v prostředí MATLAB.

Pro implementaci byla vybrána metoda inpaintingu založená na frameletech, popsaná v [5]. Funguje na obdobném principu jako algoritmus popsaný v sekci 2.5, který se liší zejména použitým typem transformace. Z toho důvodu byly naprogramovány vlastní funkce vykonávající analýzu i syntézu obrazu použitím Framelet, resp. inverzní Framelet transformace. Také byly napsány pomocné funkce pro práci s koeficienty transformace. Dále byla jako alternativa pro Framelet analýzu a syntézu obrazu použita Contourlet transformace, která byla realizována použitím nástroje Contourlet Toolbox viz [19]. Podrobnější popis procesu doplnění obrazu se nachází v sekci 3.1.

Pro snazší obsluhu algoritmu bylo v prostředí MATLAB vytvořeno GUI, které umožňuje načíst vstupní obraz a masku oblasti, jež má být doplněna. Další možností napsaného GUI je tvorba i následná editace masky, definující chybějící oblasti v obraze. Vytvořenou masku lze uložit do souboru pro pozdější využití, či ji přímo použít. Hlavním účelem navrženého GUI je přesto aplikace algoritmu, vykonávajícího samotné doplnění chybějících oblastí. Uživateli je umožněno měnit vnitřní parametry výpočtu, případně uložit výsledek výpočtu do souboru ve formátu PNG.

Další možností GUI je funkce automatického snižování hodnoty prahu, kdy algoritmus podle potřeby snižuje úroveň prahu. Hodnota prahu je snížena o její uživatelem definovanou část v případě, že se hodnoty PSNR předchozí a současné iterace rovnají. Obraz i nejdůležitější hodnoty algoritmu z iterace, v níž došlo ke snížení hodnoty prahu, jsou zaznamenány do paměti. Tato data jsou přístupná v okně *Pokročilé informace*, přičemž je možno si nechat zobrazit výsledek doplnění obrazu libovolné zaznamenané iterace. Všechny hodnoty lze vyexportovat do souboru ve formátu XLS, jednotlivé libovolné obrazy lze ukládat do souborů ve formátu PNG.

Také byla pro GUI vytvořena funkce, která uživateli promítne mapu koeficientů pro zadanou úroveň transformace. Červená barevná složka vstupního obrazu je rozložena na koeficienty, které jsou následně úhledně uspořádány a zobrazeny uživateli. Tento výstup lze uložit do souboru ve formátu PNG. Vytvořené GUI je podrobněji popsáno v sekci 3.3.

Poslední sekce této kapitoly se věnuje samotným výsledkům doplnění obrazu. Jednak jsou zde zobrazeny a komentovány dosažené výsledky doplnění obrazu, jednak se zde popisují další dosažené poznatky.

3.1 Implementace metody

V rámci této práce byl naprogramován algoritmus, který je matematicky popsán v sekci 2.5. Výpočet chybějících částí obrazu by měl probíhat v iteracích dokud nekonverguje, avšak v této implementaci z praktických důvodu probíhá výpočet v tolika iteracích, kolik zadá uživatel. Každá barevná složka obrazu je počítána zvlášť v rámci každé iterace. Jedna iterace probíhá následujícím způsobem:

1) Na vstup je v každé iteraci přiveden výsledný obraz iterace předchozí. Jedná-li se o první iteraci, vstupním obrazem je oblast definovaná pomocí Φ.

2) Obraz je rozložen na koeficienty pomocí vybraného typu transformace (Framelet nebo Contourlet), přičemž počet úrovní n transformace ovlivňuje, do jaké hloubky oblasti Ω bude pronikat informace. Vyšší úroveň transformace efektivně odpovídá delším filtrům a má za následek hlubší průnik doplňované informace do neznámé oblasti.

3) Veškeré takto vzniklé koeficienty jsou pro další zpracování uloženy do vektoru, jehož délka odpovídá počtu všech koeficientů. Hodnoty jednotlivých koeficientů jsou normovány do rozsahu $\langle 0, 1\rangle$, což je provedeno vynásobením všech jednotlivých koeficientů převrácenou hodnotou velikosti největšího koeficientu.

4) Na tento vektor koeficientů je aplikováno měkké prahování. Jedním z parametrů algoritmu je hodnota prahu `thre`, která je normovaná do rozsahu $\langle 0, 1\rangle$.

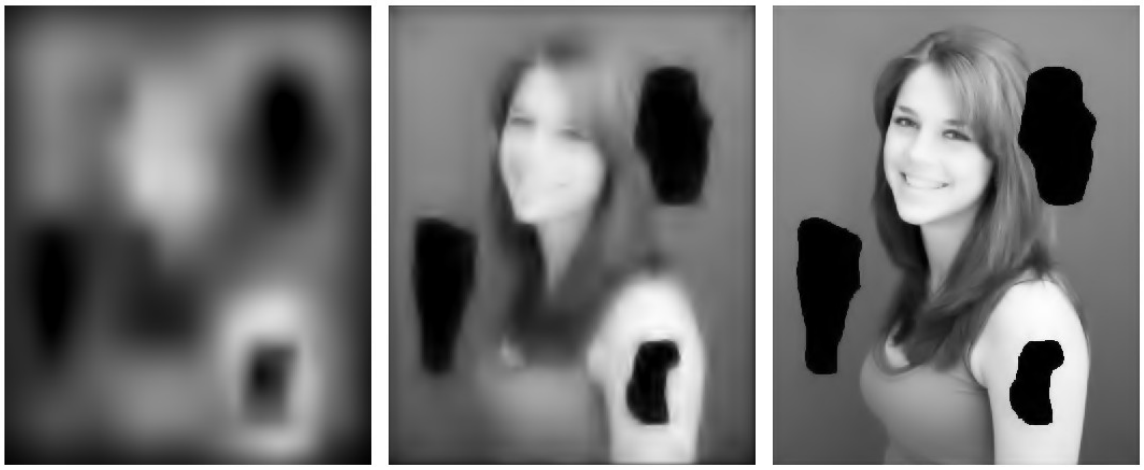
5) Hodnotám koeficientů je navrácen původní rozsah vynásobením jejich předchozí nejvyšší hodnotou použitou v bodě 3). Koeficienty jsou také opět uspořádány dle použité transformace.

6) S koeficienty se provede daný typ inverzní transformace. Tím je získán obraz, z nějž je patrné, že došlo k protažení informace do neznámé oblasti.

7) Vzniklý obraz je vynásoben inverzní maskou, čímž je získána obrazová informace v oblasti Ω. Výsledný obraz vznikne sečtením oblastí Ω a Φ.

Při vyšších hodnotách prahu (ale menších než 0,2) dochází ve výsledku k odstranění detailů a k rozmazání obrazu, jak je demonstrováno na obrázku 3.1. To napomáhá zaplnění neznámé oblasti základní informací. Naopak při nízkých hodnotách prahu se do neznámé oblasti dostávají detaily, avšak nižších kmitočtových složek sem proniká méně.

Proto se osvědčilo postupné snižování hodnoty prahu podle [10]. Nejdříve se oblast Ω zaplní základní informací a se snižující se úrovní prahu se do této oblasti postupně protlačují vyšší detaily. Informace bude ovšem pronikat do neznámé oblasti Ω jen za předpokladu, že použitá transformace je nadbytečná. Tzn. celkový počet koeficientů po transformaci je větší, než počet pixelů v obraze. Nadbytečnost závisí na typu použité transformace a na hloubce rozkladu. Při víceúrovňové Contourlet transformaci s rostoucí hloubkou dekompozice $n \to \infty$ se počet koeficientů blíží k 4/3 bodů vstupního obrazu, u Framelet transformace je to přibližně 5,3 násobek bodů obrazu.



Obr. 3.1: Vliv hodnoty prahu `thre` na výsledný obraz. Vlevo je hodnota thre = 0,08, uprostřed thre = 0,01 a vpravo thre = 0,001.

3.2 Realizace Framelet transformace

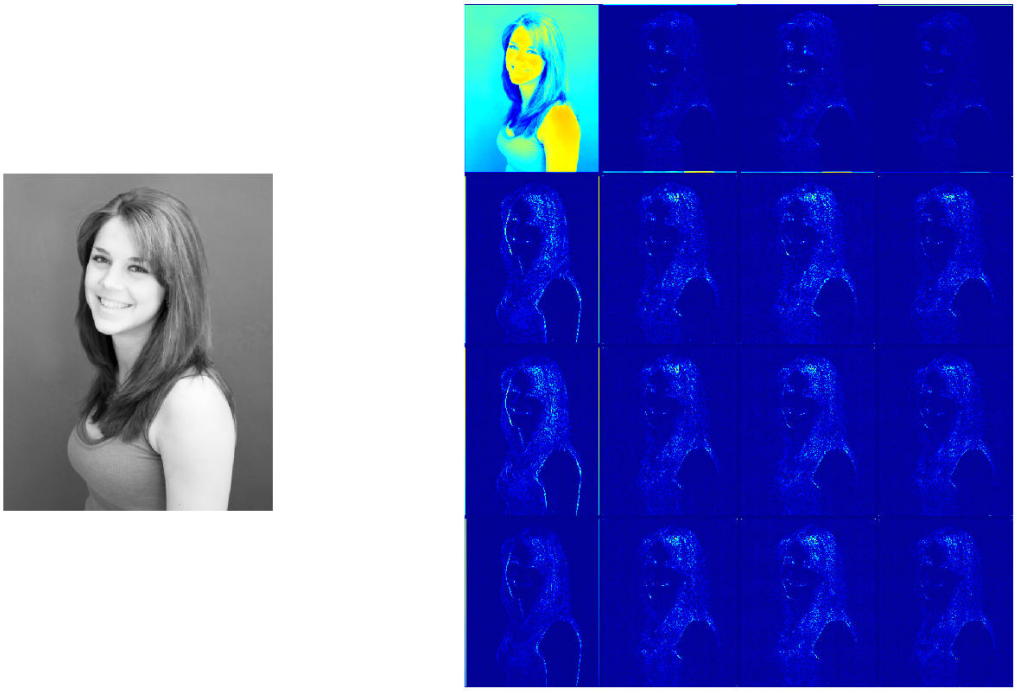
V rámci této práce bylo naprogramováno vlastní funkční řešení Framelet transformace dle [5], které bylo následně využito k doplnění obrazu. Z důvodu funkčnosti algoritmu popsaného v předchozí sekci bylo také nutné naprogramovat několik funkcí, zpracovávajících zadané úkony. Funkce jsou psány velmi obecně a nedělají jim problém ani obrazy jakýchkoliv rozměrů, ani odlišný počet filtrů, či jejich délka. Zde jsou popsány jednotlivé funkce.

1. Funkce `rozklad()` slouží k rozložení zadaného obrazu na koeficienty. Je tedy provedena Framelet transformace o n úrovních. Princip samotného rozkladu lze vysvětlit na první úrovni transformace.

Nejdříve je provedena diskrétní konvoluce vstupního obrazu a jednotlivých Framelet filtrů v horizontálním směru, řádek po řádku. Po provedení konvoluce je na její výsledek aplikováno podvzorkování dvěma. Vybraná banka filtrů obsahuje čtyři filtry, proto jsou tímto krokem vytvořeny čtyři matice koeficientů se zredukovaným počtem sloupců oproti vstupnímu obrazu.

S každou z těchto matic koeficientů i se stejnou bankou filtrů je opět provedena stejná diskrétní konvoluce, tentokráte ve vertikálním směru (sloupec po sloupci). V tomto směru dojde také k podvzorkování dvěma. Výsledkem je šestnáct matic koeficientů pro jednu úroveň transformace.

Protože první filtr z banky Framelet filtrů plní úlohu dolní propusti, proto matice koeficientů na pozici $\{1, 1\}$ slouží v případě víceúrovňové transformace jako vstupní obraz pro další úroveň rozkladu. V takovém případě tato matice nemá po již zmíněném úkonu využití a v dalších funkcích se s ní nepracuje. Výstupem funkce je buňkové pole ve tvaru sloupce o délce počtu úrovní transformace. V každé z této buněk se nachází pole matic koeficientů pro danou úroveň rozkladu. Na obrázku 3.2 se nachází zobrazení jedné úrovně rozkladu.



Obr. 3.2: Provedení jedné úrovně Framelet transformace. Vpravo je původní obraz. Vpravo se nachází vzniklé koeficienty, na které byla aplikována mapa barev ”jet”.

2. První pomocná funkce je `koeficient_na_vektor()`, která uspořádá všechny užitečné koeficienty, vzniklé ve funkci rozkladu, do vektoru o délce rovné jejich součtu. Výstupem funkce je jednak již zmíněný vektor, jednak matice vyjadřující strukturu koeficientů. Tato matice struktury obsahuje pět sloupců. První sloupec udává úroveň rozkladu, druhý a třetí sloupec značí pozici matice koeficientů a poslední dva sloupce vyjadřují rozměry této matice. Každý řádek vyjadřuje jednu matici koeficientů. Bez této struktury není možné později uspořádat koeficienty z vektoru zpět na své místo.

3. Pomocná funkce `vektor_na_koeficient()` provádí operaci inverzní k funkci předchozí. Funkce pomocí matice struktury uspořádá jednotlivé koeficienty ze vstupního vektoru zpět do buněk úrovní a matic koeficientů. Jedná se o stejné rozvržení, k jakému došlo na výstupu funkce `rozklad()`.

4. Funkce `kompozice()` provede inverzní Framelet transformaci, čímž se vykoná rekonstrukce obrazu z jednotlivých matic koeficientů. Jelikož je tato funkce inverzní k funkci `rozklad()`, je nutno postupovat v opačných krocích. Postupuje se od nejvyšší úrovně transformace k nejnižší. Princip lze jednoduše popsat na nejvyšší úrovni rozkladu.

   Pole matic koeficientů si lze představit jako pole buněk o rozměrech 4×4, kde číslo 4 znamená počet filtrů vybrané Framelet banky. Nejdříve se provede vertikální nadvzorkování dvěma první matice koeficientů, aktuálního řádku pole buněk dané úrovně. Poté následuje diskrétní konvoluce výsledku nadvzorkování a obráceného příslušného filtru Framelet banky, taktéž ve vertikálním směru. To se provede také pro všechny ostatní matice koeficientů dané úrovně. Matice koeficientů pole buněk se sečtou po řádcích, tím vzniknou čtyři matice o počtu řádků shodném s koeficienty nižšího řádu transformace. Nad každou z těchto matic je provedeno nadvzorkování dvěma v horizontálním směru (řádek po řádku) a následná konvoluce tohoto výsledku a obráceného příslušného filtru banky frameletů. Pak se tyto čtyři zbývající matice sečtou a jedná-li se o nejnižší úroveň transformace, pak jde o výsledný obraz. Existuje-li o číslo nižší úroveň rozkladu, pak je tento výsledek uložen na místo $\{1, 1\}$ pole buněk dané úrovně a celý algoritmus se opakuje.

5. Funkce `zobraz_koeficienty()` je poslední vytvořenou funkcí, jež je spřízněna s Framelet transformací. Funkce slouží k přehlednému zobrazení koeficientů, vzniklých v průběhu rozkladu obrazu pomocí Framelet transformace.

3.3 Popis grafického uživatelského rozhraní

Uživatelské rozhraní se spouští pomocí souboru aplikace_metody.m. V jeho okně lze pomocí menu *Okno* přepínat mezi třemi formuláři, vyjadřujícími hlavní zobrazení, tvorbu masky a pokročilé informace. Ovládací prvky v těchto formulářích jsou vybaveny skrytou nápovědou, jež se zobrazí při delším pobytu kurzoru v jejich oblasti.

3.3.1 Hlavní zobrazení

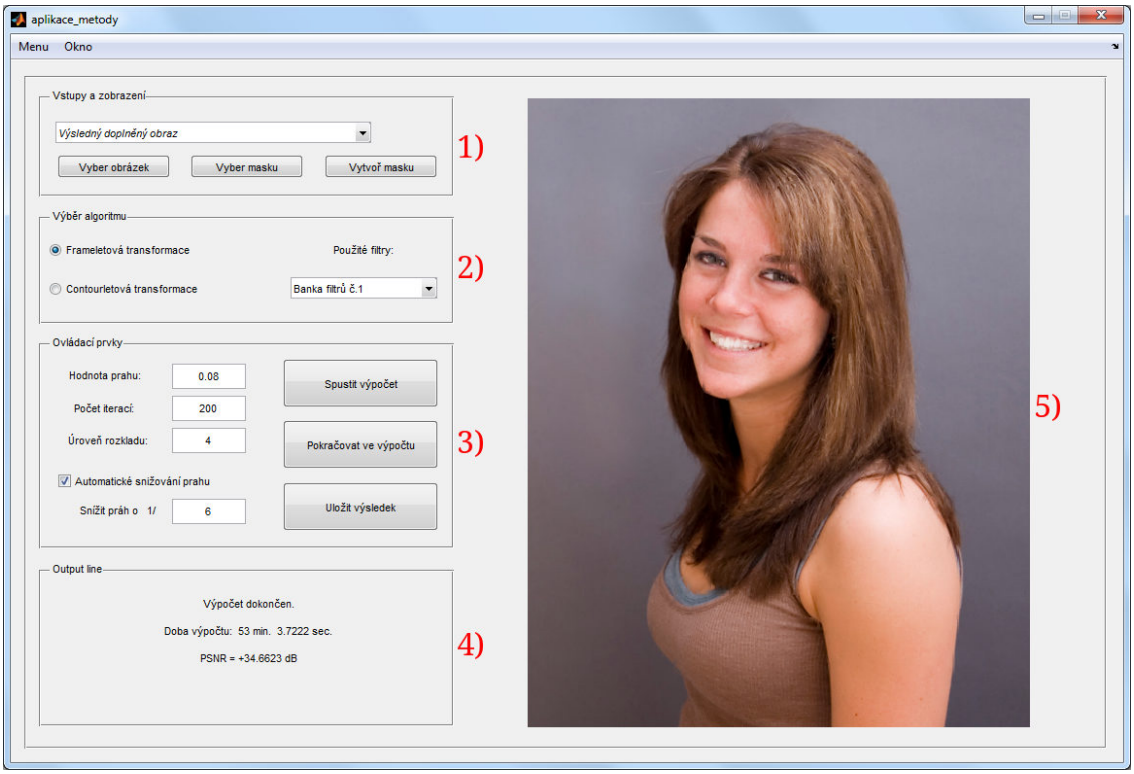
Vzhled hlavního zobrazení navrženého GUI je zobrazen na obrázku 3.3. Jak je zvýrazněno na obrázku, tak lze rozdělit hlavní okno do následujících čtyř oblastí:

1) **Vstupy a zobrazení** slouží k manipulaci se vstupy výpočtu a k možnosti jejich zobrazení. Pomocí pop-up menu je uživateli umožněno nechat si zobrazit vstupní obraz I, masku Ω , oblast Φ , případně výsledek doplnění na plátně pro zobrazení vpravo.

   Dále se zde nacházejí tři tlačítka, jejichž alternativu lze nalézt v hlavním menu. Jejich funkcí je:

   (a) *Vyber obrázek.* Pomocí tlačítka lze vybrat a načíst vstupní obraz I, obvykle ve formátu PNG či JPG.
   (b) *Vyber masku.* Tlačítko slouží k vybrání a načtení masky Ω ze souboru ve formátu PNG. Maska je tvořena monochromatickým obrazem a nabývá stejných rozměrů jako vstupní obraz. Pixelům, jež jsou považovány za chybějící je přiřazena hodnota 0 (černá barva), ostatní pixely mají hodnotu 255 (bílá barva). Pixel s jinou hodnotou je uvažován jako bílý.
   (c) *Vytvoř masku.* Po spuštění se započne uživatelská tvorba nové masky, také se změní formulář hlavního zobrazení na tvorbu masky. Proces je dále popsán v podsekci 3.3.2.

2) **Výběr algoritmu.** Pro účely doplnění obrazu lze zvolit mezi frameletovou a contourletovou transformací. Rozklad a syntéza pomocí frameletové transformace byly naprogramovány i s dalšími pomocnými funkcemi v rámci bakalářské práce, s tímto výběrem je svázáno pop-up menu pro výběr banky filtrů. Naopak contourletová transformace je řešena použitím nástroje Contourlet Toolbox. Pro jeho použití je ovšem nutno zadat vstupní obraz i masku o rozměrech dělitelných faktorem podvzorkování při zadaném počtu úrovní transformace.



Obr. 3.3: Vzhled formuláře hlavního zobrazení realizovaného GUI.

3) **Ovládací prvky.** V tomto panelu lze upravovat základní parametry algoritmu mezi něž patří počet úrovní transformace, hodnota prahu, počet provedených iterací výpočtu či velikost snižování prahu.

   Tlačítkem *Spustit výpočet* se zahájí proces doplňování neznámých částí obrazu. Během samotného výpočtu sice nelze s GUI interagovat, uživatel je ale informován o průběhu výpočtu několika způsoby. Jednak je aktuální stádium výpočtu zobrazeno pomocí progressbaru, dále je vyjádření aktuálních hodnot vypsáno v informačním poli a navíc se na plátně v bloku 4) zobrazuje výsledek aktuální iterace výpočtu. Doba výpočtu především závisí na počtu nastavených iterací. Po skončení výpočtu je uživatel informován o době jeho trvání a je zobrazena hodnota špičkového poměru signálu k šumu (PSNR) výsledku oproti původnímu vstupnímu obrazu.

   Tlačítko *Pokračovat ve výpočtu* spustí stejnou funkci pro výpočet jako předchozí tlačítko, přičemž vstupním obrazem je výsledek předchozího výpočtu. Doplněný obraz lze pomocí sousedního tlačítka *Uložit výsledek* zaznamenat do souboru ve formátu PNG.

Checkboxem *Automatické snižování prahu* je aktivována funkce postupného snižování hodnoty prahu. Po každé iteraci je spočtena hodnota PSNR a rovná-li se hodnotě PSNR z předchozí iterace, je hodnota prahu snížena o Δ_{thre} hodnoty prahu. Hodnotu Δ_{thre} může uživatel nastavit o položku níže.

4) **Informační pole** Smyslem informačního pole je zobrazovat uživateli aktuální informace, týkající se průběhu výpočtu, případně výjimečných stavů.

5) **Plátno pro zobrazení.** V posledním objektu formuláře hlavního zobrazení se uživateli promítá výsledek doplnění obrazu nebo vstupy algoritmu, mezi kterými může uživatel přepínat v bloku 1).

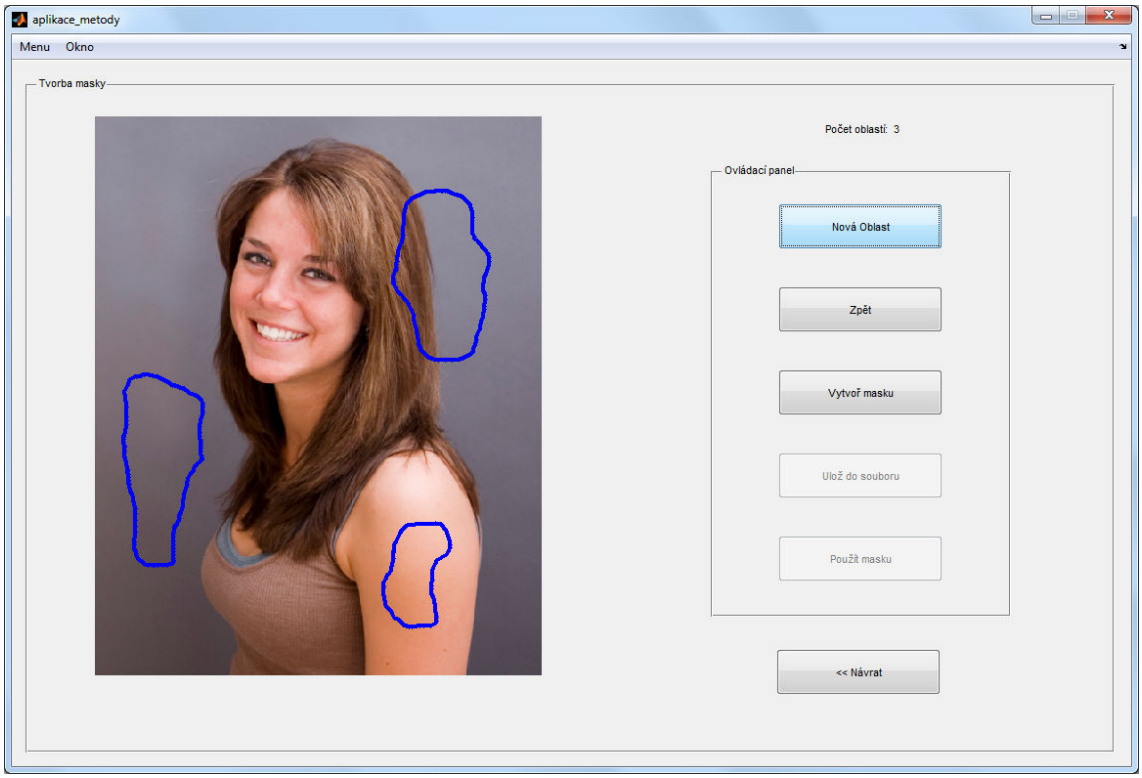
3.3.2 Tvorba masky

Realizované GUI umožňuje uživateli definovat oblasti, které jsou považovány za chybějící. Uživateli je tato funkce zpřístupněna po stisknutí tlačítka *Vytvoř masku,* nalézajícího se v panelu *Vstupy a zobrazení* či v horním Menu. Tímto způsobem se začíná práce na nové masce a starší tvorba masky bude vymazána. Formulář tvorby masky je zobrazen na obrázku 3.4. Mezi okny lze přepínat v horním menu Zobrazení, aniž by došlo ke zrušení právě tvořené masky.

Jednotlivé funkce tvorby masky jsou uživateli k dispozici v ovládacím panelu vpravo. Po inicializaci tvorby nové masky je zde uživateli k dispozici jediné tlačítko *Nová oblast* a ostatní tlačítka nejsou přístupná.

Po stisknutí tlačítka *Nová oblast* se spustí funkce `sledovani_mysi()`, jež umožní uživateli definovat oblast k doplnění. Tato funkce vznikla přepracováním nástroje Freehand Contour Drawing [20]. Uživatel přesune kurzor myši na plátno s obrazem vlevo nad požadovanou oblast a podržením levého tlačítka myši i posouváním myši začne kreslit body obrysu požadované oblasti. Výběr oblasti je ukončen po uvolnění levého tlačítka myši. Vybrané body se propojí silnou modrou čárou, přičemž se naváže koncový bod na počáteční. Stejným způsobem může uživatel přidávat další oblasti, jejich počet je však omezen na sto. Počet vytvořených oblastí je zobrazován v informačním řádku vpravo nahoře.

Uživatel si může povšimnout, že mu byla zpřístupněna tlačítka *Zpět* a *Vytvoř masku*. Tlačítkem *Zpět* se odstraňuje poslední vytvořená oblast a opakovaným stiskem se dají postupně odebrat všechny oblasti. Po odstranění poslední vytvořené oblasti se opět obě tlačítka znepřístupní.



Obr. 3.4: Vzhled formuláře tvorby masky realizovaného GUI.

Druhé zmíněné tlačítko *Vytvoř masku* nahradí obsah vytvořených oblastí černými pixely (s hodnotou 0) a zbylé pixely budou bílé (hodnota 255). Uživateli se odemknou poslední dvě tlačítka *Ulož do souboru* a *Použít masku*. Stále je však možné se vrátit do módu editace masky stiskem tlačítka *Nová oblast* nebo *Zpět*.

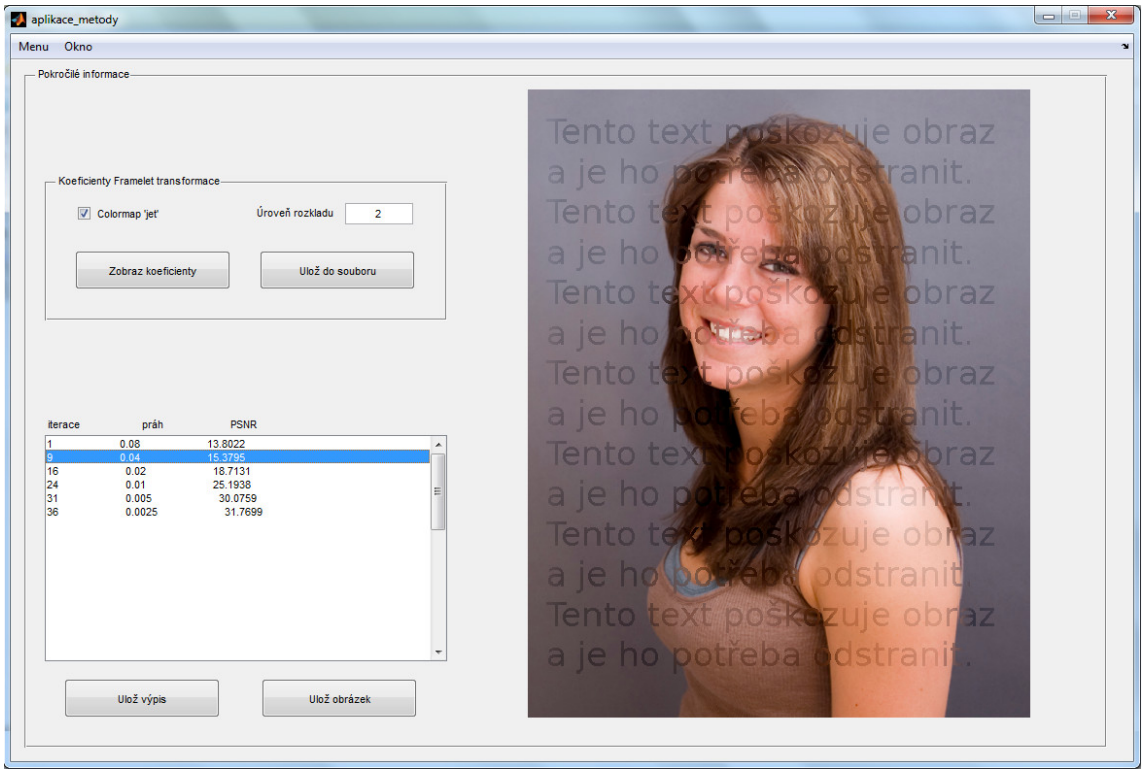
Pomocí tlačítka *Ulož do souboru* lze uložit masku do souboru ve formátu PNG pro pozdější využití. Pomocí tlačítka *Použít masku* se přepne pohled na formulář hlavního zobrazení a na vstup algoritmu definující oblast k doplnění je přivedena vytvořená maska. K navrácení do formuláře hlavního zobrazení, aniž by došlo k použití vytvořené masky, slouží tlačítko *Návrat*.

3.3.3 Pokročilé informace

Poslední okno slouží zejména k tomu, aby si uživatel mohl prohlédnout zaznamenaný průběh doplnění obrazu. Z praktických důvodu dochází k takovému záznamu jen v případě, že byla během výpočtu aktivní volba automatického snižování prahu. Toto informativní okno je ukázáno na obrázku 3.5. Jak lze pozorovat, tak se dá toto okno rozdělit do tří samostatných oblastí. První oblastí, které si uživatel všimne je plátno pro zobrazení vpravo. Jeho účelem je promítnout uživateli požadovaný obraz.

Další samostatná oblast se nachází vlevo nahoře a zabývá se zobrazením mapy koeficientů Framelet rozkladu na zobrazovací ploše první oblasti. Uživatel má možnost nastavit si úroveň rozkladu pomocí stejnojmenně označené položky. Pro případ, že by byly hodnoty koeficientů malé, a tedy špatně viditelné, je zde k dispozici checkbox pro aplikování mapy barev „jet“. Dále se zde nachází tlačítko *Zobraz koeficienty*. To nejdříve provede rozklad červené složky vstupního obrazu pomocí funkce `rozklad()`, a poté pomocí funkce `zobraz_koeficienty()` koeficienty úhledně uspořádá a zobrazí na ploše v první oblasti. Vytváření mapy koeficientů může chvíli trvat. Proto se mapa koeficientů počítá během prvního stisknutí tlačítka, v případě že byla změněna úroveň rozkladu, nebo pokud byla změněna možnost palety barev „jet“. Výsledný obraz lze uložit do souboru ve formátu PNG.

Poslední oblast se nachází vlevo dole a stojí za vznikem tohoto okna. Nalézá se tu listbox, ve kterém jsou zobrazeny zaznamenané parametry průběhu doplnění, a to číslo dané iterace, úroveň prahu a hodnota PSNR. Uživatel může tato data procházet a po kliknutí na řádek zaznamenané iterace je na zobrazovací ploše vpravo zobrazen výsledek dané iterace. Tento výpis lze vyexportovat do souboru ve formátu XLS pomocí tlačítka *Ulož výpis*. Tlačítkem *Ulož obrázek* lze naopak uložit výsledek doplnění vybrané iterace do souboru typu PNG.



Obr. 3.5: Okno pokročilých informací realizovaného GUI.

3.4 Výsledky algoritmu

V této sekci jsou popsány výsledky doplnění obrazu, kterých bylo dosaženo v rámci bakalářské práce. Pro testování algoritmu doplnění obrazu byl vybrán obrázek, který oplývá jak většími volnými plochami s minimální změnou obrazové informace, tak oblastmi s texturou a strukturou. To umožňuje zobrazit různé jevy, vznikající při doplnění chybějících částí obrazu danou metodou. Tento obrázek také disponuje volnější licencí a jeho použití pro tyto účely není v rozporu s autorskými právy.

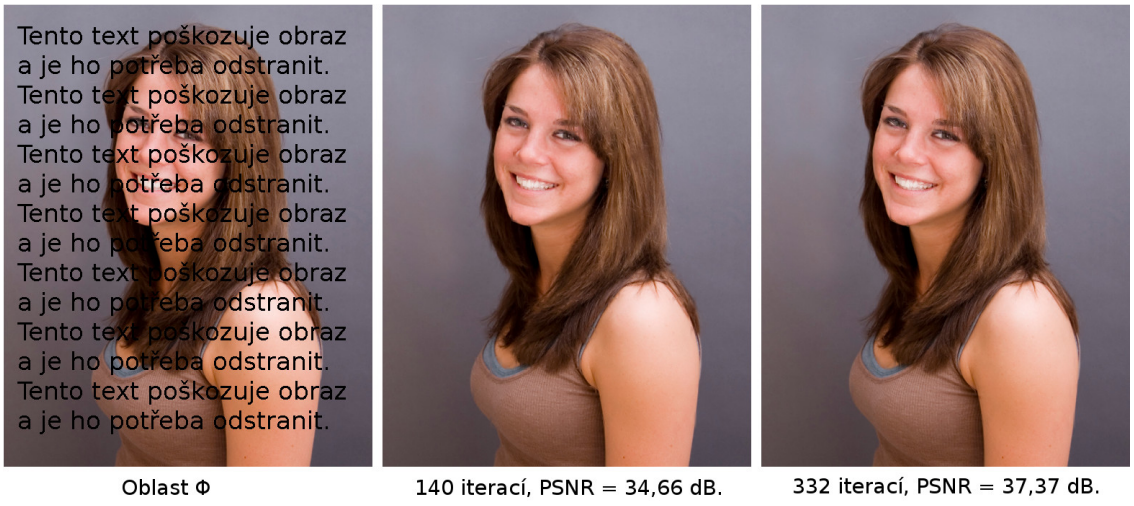
Průběh samotného doplnění obrazu si lze prohlédnout na obrázku 3.6. Pro doplnění obrazu bylo použito algoritmu s Framelet transformací a postupným snižováním hodnoty prahu `thre`. Zobrazení od 24. iterace jsou zde zachycena vždy v okamžiku před snížením hodnoty prahu.

Obr. 3.6: Průběh doplnění obrazu za použití Framelet transformace. Od 140. iterace se hodnota PSNR neměnila, ani nebyly pozorovány rozdíly v obraze.

Rozdíl prahu Δ_{thre}, neboli hodnota o kterou se automaticky snižuje práh `thre`, byla nastavena ve všech výpočtech zobrazených v této sekci na hodnotu 1/6. Pro rychlejší průběh výpočtu lze ovšem nastavit vyšší hodnotu rozdílu prahu. Experimentálně však bylo zjištěno, že pokud se hodnota prahu snižuje o vyšší Δ_{thre}, tak může dojít k o něco horšímu výslednému doplnění. Jestliže byla hodnota rozdílu prahu zvýšena na 1/2 , pak výsledná dosažená hodnota PSNR klesla o 0,08 dB. Rozdíl je to však nepatrný a jen slabě pozorovatelný na rozhraní dvou a více oblastí. Počet iterací potřebných k doplnění obrazu klesl ze 140 na 52.

Z hlediska diverzity výsledků a možnosti jejich porovnání byly v použitém algoritmu (také popsaném v sekci 3.1) postupně aplikovány dva typy transformace. Prvním typem je Framelet transformace, jejíž řešení bylo naprogramováno exkluzivně pro účely této bakalářské práce. Druhým typem je Contourlet transformace, jež byla řešena za použití nástroje Contourlet Toolbox.

Oba způsoby jsou poprvé porovnány na obrázku č. 3.7, kde bylo doplněno úzké písmo nacházející se v obraze. Pro doplnění užších oblastí byly použity 4 úrovně dekompozice. Počet směrů Contourlet banky podle úrovní byl [4 4 6 6] .



Obr. 3.7: Porovnání výsledku doplnění velmi úzkých oblastí (vlevo) pomocí Framelet transformace (uprostřed) a Contourlet transformace (vpravo).

Výsledky se však liší v detailech, jak je promítnuto na obrázku 3.8. Způsob s Framelet transformací se příliš nehodí k doplnění náročnější textury, jak lze vidět v oblasti vlasů. To naopak nedělá problém způsobu s Contourlet transformací. Žádný z těchto způsobů si neporadí s oblastí, kde dochází na malé ploše k velkým změnám obrazové informace. Toho si lze všimnout obrysem písmene „b“ nad pravým okem.

Obr. 3.8: Detail porovnání výsledku doplnění oblastí definovaných úzkým písmem pomocí Framelet transformace (vlevo) a Contourlet transformace (vpravo).

Na obrázku 3.9 je zobrazen výsledek doplnění rozsáhlejších oblastí, definovaných tučným velkým písmem. Z výsledků je patrné, že doplnění nedopadlo zdaleka tak ideálně, jako v předchozím případě. Při použití Contourlet transformace opět došlo k lepšímu doplnění textury (oblast vlasů). Také byly načrtnuty hrany, byť oblasti poblíž hran nebyly uspokojivě doplněny. Algoritmu s Framelet transformací se nepodařilo přijatelně obnovit texturu a hrany, naopak se mu lépe vedlo v oblastech bez struktury a poblíž hran.

Obr. 3.9: Porovnání doplnění oblastí definovaných silným písmem (vlevo) pomocí Framelet transformace (uprostřed) a Contourlet transformace (vpravo).

Předností použité Framelet transformace je vytvoření velkého množství nadbytečné informace, tedy počet koeficientů rozkladu je daleko větší než počet bodů v původním obraze. V porovnání s Contourlet transformací je proto zapotřebí méně kroků algoritmu k doplnění obrazu, byť jedna iterace je výpočetně náročnější.

V předchozích příkladech bylo použito čtyř úrovní rozkladu. Pro doplnění rozsáhlých oblastí tato hloubka transformace nestačí, resp. k samotnému zacelení takové oblasti by bylo potřeba mnoho tisíc iterací. Pro větší hloubku vniku informace do neznámé oblasti, definované pomocí Ω, je nutné využít více úrovní rozkladu. Větší hloubka transformace odpovídá delším filtrům. Na obrázku 3.10 se nachází porovnání výsledků doplnění rozsáhlých oblastí. V obou případech bylo použito šest úrovní rozkladu. Počet směrů Contourlet banky podle úrovní byl $[1\ 2\ 4\ 4\ 6\ 6]$. Oblasti bez textury a struktury byly uspokojivě doplněny jen za pomoci Framelet transformace. Contourlet transformace nevytváří dostatečné množství nadbytečné informace.

Obr. 3.10: Porovnání doplnění rozsáhlých oblastí pomocí Framelet transformace (uprostřed) a Contourlet transformace (vpravo).

Nakonec je důležité zmínit výrazný rozdíl v časové náročnosti výpočtu jedné iterace obou implementovaných transformací. Funkce analýzy a syntézy obrazu jsou zde časově zdaleka nejnáročnějšími úkony. Všechny funkce pracující s Framelet transformací byly napsány v prostředí MATLAB, což se negativně projevuje na výkonu. Naopak autoři nástroje Contourlet Toolbox v něm tyto funkce analýzy a syntézy obrazu optimalizovali přepsáním do jazyka C. Menší počet koeficientů Contourlet transformace znamená i nižší časové nároky na operace spojené s měkkým prahováním. Na počítači s procesorem Intel Core2Duo @2000 MHz trvala jedna iterace doplnění obrázku o rozměrech 512×640 při použití Contourlet transformace průměrně 2,9 s. Jedna iterace s Framelet transformací zde trvala průměrně 46,1 s.

4 ZÁVĚR

Cílem této práce bylo jednak zpracování přehledu a zhodnocení moderních metod pro doplnění chybějících částí obrazu, jednak vybranou metodu implementovat jako funkci v prostředí MATLAB. Dalšími cíli bylo vytvořit v prostředí MATLAB GUI s dostatečnou úrovní interakce a zhodnotit účinnost a výpočetní náročnost implementovaného algoritmu.

Poté, co byly v první kapitole vysvětleny a popsány některé pojmy používané dále v této práci, následovalo rozdělení existujících metod podle přístupu a popsání jednotlivých metod. Poté, co byla každá metoda uvedena, následoval popis jejího algoritmu a zhodnocení dle poznatků, nabytých z dostupné literatury.

Metoda Image Inpainting je jednou z nejstarších metod pro doplňování chybějících částí obrazu. Tato metoda je vhodná pro opravení škrábanců a jiných podobně úzkých oblastí. Naopak si dobře neporadí s doplněním textury, hran a rozsáhlých oblastí. Z důvodu potřeby doplnění textury vznikla metoda Fragment-Based Image Completion, založená na přístupu syntézy textury. Nachází-li se v obraze dostatek potřebné informace, tak je tato metoda schopná doplnit jak chybějící texturu, tak i jednoduché hrany. Jedná se však o výpočetně velmi náročnou proceduru a se zvyšujícím se rozlišením obrazu roste čas potřebný k doplnění obrazu exponenciálně.

Metoda Exemplar-Based Image Inpainting byla vyvinuta pro doplnění rozsáhlých oblastí v obraze. Funguje na principu protahování textury i struktury pomocí kopírování bloků (patch). Tato metoda dokáže v případě jednoduššího uspokojivě doplnit velké oblasti v celkem krátkém čase. Problémy jí může dělat doplnění obrazu se složitou strukturou. Podobný přístup, založený na kopírování bloků, používá i metoda nazvaná Gradient-Based Image Completion by Solving Poisson Equation. Výsledkem tohoto náročnějšího algoritmu je lepší doplnění rozsáhlých oblastí obrazu se složitou strukturou. Nevýhodou algoritmu je horší doplnění zakřivených hran a nejednoznačných struktur.

Nakonec byla popsána metoda Inpainting by Flexible Haar-Wavelet Shrinkage. Tato metoda inpaintingu staví na myšlence anizotropické regularizace a difúze. Účelem této metody je doplnit úzké oblasti typu šrámů nebo písma, naopak se příliš nehodí pro doplnění rozsáhlé oblasti. Také jí dělá problémy doplnění hrany. Na stejném principu funguje i algoritmus inpaintingu, založený na Framelet transformaci, který byl implementován v rámci praktické části této bakalářské práce.

Pro použití v algoritmu, napsaném v rámci praktické části bakalářské práce, bylo také naprogramováno funkční řešení Framelet transformace. Jedná se o funkce obstarávající analýzu i syntézu obrazu a pomocné funkce realizující převod mapy koeficientů rozkladu na vektor a zpět, či funkce pro zobrazení mapy koeficientů.

Dle zadání bylo vytvořeno grafické uživatelské rozhraní (GUI). To poskytuje uživateli odpovídající možnost interakce a navíc jsou všechny ovládací prvky a vstupy vysvětleny pomocí skryté nápovědy. Uživatel je informován jak o průběhu i výsledku výpočtu, tak o nestandardních situacích. V průběhu výpočtu je uživatel zpravován o aktuálním číslu iterace, velikosti prahu a hodnotě PSNR. Na konci každé iterace je zobrazen aktuální výsledek doplnění neznámých oblastí v obraze. Prohlížení i uložení zaznamenaných dílčích kroků výpočtu je uživateli umožněno v případě, že byla vybrána možnost automatického snižování hodnoty prahu. Také si lze nechat zobrazit mapu koeficientů rozkladu pomocí Framelet transformace. Navíc byl vytvořen nástroj, kterým lze pomocí myši vytvořit masku, definující oblast Ω.

Algoritmus byl pro určitou diverzitu výsledků testován ve dvou provedeních. V prvním případě se v algoritmu použila Framelet transformace. Počet koeficientů, vznikajících při větší hloubce Framelet transformace, se blíží 5,3 násobku počtu pixelů v obraze. V druhém případě byla použita Contourlet transformace, jejíž počet koeficientů se blíží 4/3 počtu bodů obrazu. Při použití měkkého prahování tato nadbytečná informace proniká do chybějící oblasti a tím ji zaplňuje. Při vyšší hodnotě prahu se do této oblasti dostávají nižší kmitočtové složky, při nízkém prahu sem naopak pronikají jen vyšší kmitočtové složky. Proto se osvědčilo využívat postupné snižování prahu. Z toho vyplývá, že algoritmus s Framelet transformací potřebuje daleko méně iterací k doplnění obrazu, než algoritmus s Contourlet transformací.

Předností použití Contourlet transformace je správné doplnění textury a hran v užších chybějících oblastech. Díky relativně malé nadbytečnosti jí dělá problém jednak doplnění oblastí v těsné blízkosti velké změny charakteru obrazové informace, jednak doplnění rozsáhlých oblastí. V případě použité Framelet transformace dojde naopak v oblastech textur i hran k rozmazání. Úzké oblasti poblíž velkých změn obrazové informace jsou lépe doplněny a navíc zvládá doplnění rozsáhlých oblastí bez textury a hran.

V případě použitého řešení Contourlet Toolbox jsou nejnáročnější úkony (analýza a syntéza obrazu) výpočtu optimalizované v jazyce C. Také kvůli daleko menšímu množství koeficientů rozkladu trvalo algoritmu zpracovat jednu iteraci doplnění obrazu o rozměrech 512×640 průměrně 2,9 s na počítači s procesorem Core2Duo @2000 MHz. Díky tomu, že byla Framelet transformace napsána v pomalejším prostředí MATLAB, a také kvůli ohromnému množství koeficientů trvala stejná iterace průměrně 46,1 s. Počet iterací, potřebných k doplnění obrazu, závisel jak na použité transformaci, tak na doplňované oblasti, tak na nastavení parametru rozdílu prahu.

LITERATURA

[1] ABDELNOUR, A. F. Dense grid framelets with symmetric lowpass and bandpass filters. *International Symposium on Signal Processing and Its Applications.* Únor 2007, roč. 9, s. 1-4.

[2] ALVAREZ, L.; LIONS, P.-L.; MOREL, J.-M. Image Selective Smoothing and Edge Detection by Nonlinear Diffusion II. *SIAM Journal on Numerical Analysis.* Červen 1992, roč. 29, č. 3, s. 845-866.

[3] BARNES, C., et al. PatchMatch : A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (TOG).* Srpen 2009, roč. 28, č. 3, s. 24.

[4] BERTALMIO, M.; SAPIRO, G.; BALLESTER, C.; aj. Image Inpainting. *SIGGRAPH*, 2000: s. 417–424.

[5] CAI , J.-F.; CHAN , R. H.; SHEN , Z. A framelet-based image inpainting algorithm. *Applied and Computational Harmonic Analysis.* 2008, ročník 24, číslo 2, s. 131-149.

[6] CHAN, R.H.; SETZER, S.; STEIDL, G. Inpainting by Flexible Haar-Wavelet Shrinkage. *SIAM Journal on Imaging Sciences.* Červenec 2008, ročník 1, č. 3, s. 273-293.

[7] CRIMINISI, A.; PERÉZ, P.; TOYAMA, K. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Transactions on Image Processing*, ročník 13, č. 9, září 2004: s. 1200–1212.

[8] DRORI, I.; COHEN-OR, D.; YESHURUN, H. Fragment-Based Image Completion. *ACM Transactions on Graphics,* ročník 22, č. 3, 2003: s. 303–312.

[9] GROHOL, J. *Doplňování chybějících částí obrazu.* Brno, 2009. 50 s. Diplomová práce. MUNI. Vedoucí diplomové práce RNDr. Vít Kovalčík Ph.D.

[10] GULERYUZ, O. G. Nonlinear Approximation Based Image Recovery Using Adaptive Sparse Reconstructions and Iterated Denoising: Part I - Theory. *IEEE Trans. Image Proc.* 2004, č. 15, s. 555-571.

[11] IGNÁCIO, U. A.; JUNG, C. R. Block-based image inpainting in the wavelet domain. *The Visual Computer International Journal of Computer Graphics.* Springer-Verlag 2007. s. 733-741. ISSN 01782789

[12] RAJMIC, P. Pracovní text k předmětu Základy počítačové sazby a grafiky. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 125 s.

[13] SHEN, J., et al. Gradient based image completion by solving the Poisson equation. *Computers & Graphics*. Leden 2007, roč. 31, č. 1, s. 119-126.

[14] STARCK, J.-L.; MURTAGH, F.; FADILI, J. M. *Sparse Image and Signal Processing : Wavelets, Curvelets, Morphological Diversity*. New York : Cambridge University Press, 2010. xvii, 319 s. ISBN 978-0-521-11913-9.

[15] SUN, J.; YUAN, L.; JIA, J.; SHUM, H.-Y. Image Completion with Structure Propagation. *SIGGRAPH*. 2005, č.24, s. 861-868.

[16] ŠMARDA, Z.; RŮŽIČKOVÁ, I. Skripta k předmětu Vybrané partie z matematiky. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, říjen 2004. 168 s.

[17] ZÁTYIK, J. *Směrové reprezentace obrazů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 76 s. Vedoucí semestrální práce Ing. Zdeněk Průša.

[18] ŽÁRA, J., et al. *Moderní počítačová grafika*. 2. vyd. Praha : Computer Press, 2005. 609 s. ISBN 80-251-0454-0.

Seznam použitých knihoven programu MATLAB

[19] DO, M. N. *Contourlet toolbox* [knihovna programu MATLAB]. Ver 2.0. 2003 [cit. 13.12.2011]. Dostupné z URL:
<http://www.ifp.illinois.edu/~minhdo/software/contourlet_toolbox.zip>

[20] PESHKIN, L. *Freehand Contour Drawing* [knihovna programu MATLAB]. Ver Září 2005 [cit. 15.5.2012]. Dostupné z URL:
<http://www.mathworks.com/matlabcentral/fileexchange/8262-freehand-contour-drawing>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

Obecné symboly vstupů algoritmu

I vstupní obraz

Φ známá oblast obrazu

Ω vstupní maska – definuje neznámou oblast, jež má být doplněna

$\partial\Omega$ hranice oblastí Φ a Ω

Image Inpainting

∇I^n gradientní vektor

I_t^n aktualizace obrazu I^n

$\overrightarrow{\delta L^n}$ rozsah změny informace

$\overrightarrow{N}$ směr rozšíření informace

Fragment-Based Image Completion

α inverzní maska

β mapa důvěryhodnosti

$d(\Psi_\mathrm{a}, \Psi_\mathrm{b})$ vzdálenost mezi dvěma záplatami Ψ_a a Ψ_b

v_i mapa kandidátů

Exemplar-Based Image Inpainting

C důvěryhodnost patche

D data

$\boldsymbol{n}_\mathrm{p}$ vektor kolmý na hranici $\partial\Omega$ v bodě p

P priorita patche

Ψ patch, blok dat

Gradient-Based Image Completion by Solving Poisson Equation

G gradient, zjednodušení zápisu

Inpainting by Flexible Haar-Wavelet Shrinkage

A operátor analýzy (dekompozice)

A^* operátor syntézy (rekonstrukce, kompozice)

$\mathcal{I}$ jednotková matice

Λ diagonální matice

$\varepsilon(j)$ šum

$\mathcal{T}_\Lambda$ operátor měkkého prahování

$\widehat{u}_{r+1}$ výstupní obraz dané iterace

Ostatní

$\downarrow 2$ podvzorkování dvěma

GUI grafické uživatelské rozhraní

MSE střední kvadratická chyba (mean squared error)

PNG formát souboru obrázků (Portable Network Graphics

PSNR špičkový poměr signálu k šumu (peak signal-to-noise ratio) [dB]

SSD součet čtverců rozdílů (sum of squared difference)

`thre` hodnota prahu, v algoritmu použita pro měkké prahování

Δ_{thre} rozdíl prahu, slouží k postupnému snižování hodnoty prahu `thre`

`XLS` přípona souborů specifikace Office Open XML

SEZNAM PŘÍLOH

A NÁVOD PRO SPUŠTĚNÍ GUI.

Řešení vybrané metody pro doplnění obrazu, pomocné funkce a grafické prostředí se nachází ve složce MATLAB. Realizované grafické uživatelské rozhraní lze spustit v prostředí MATLAB pomocí souboru aplikace_metody.m. Návrh grafického rozhraní lze otevřít stiskem tlačítka GUIDE, přepnutím na Open Existing GUI a výběrem souboru aplikace_metody.fig.

Pokud po prvním spuštění výpočtu doplnění obrazu prostřednictvím Contourlet transformace vyskočí chyba, je to pravděpodobně tím, že je potřeba zkompilovat soubor starající se o analýzu a syntézu obrazu. Postup je jednoduchý. Po přepnutí do složky contourlet_toolbox_tight napište v MATLABU příkaz: `mex resampc.c` poté vyberte ze seznamu dostupných kompilátorů. Pokud se nezobrazil žádný, nainstalujte si kompilátor kompatibilní s Vaší verzí prostředí MATLAB.

Dále nástroj Contourlet Toolbox umí pracovat jen s určitými obrazy. Rozměry vstupního obrazu i masky musí být dělitelné dvěma (podvzorkování) pro všechny úrovně rozkladu. Naopak realizovaná Framelet transformace dokáže zpracovat jakékoliv obrazy.

B OBSAH CD

MATLAB

- **contourlet_toolbox_tight** - složka obsahuje Contourlet Toolbox.
- **pictures** - složka obsahuje základní volitelné obrázky a masky oblasti Ω.
  - amy_lee.png
  - amy_lee_maska_01.png
  - amy_lee_maska_02.png
  - amy_lee_maska_03.png
  - fruit.png
  - fruit_maska.png
  - lena.png
  - lena_maska_01.png
  - lena_maska_02.png
- aplikace_metody.fig - soubor GUI prostředí MATLAB.
- aplikace_metody.m - soubor zdrojového kódu pro GUI, spouštěcí soubor.
- koeficient_na_vektor.m - soubor pomocné funkce pro uložení všech koeficientů rozkladu pomocí Framelet transformace do jednoho vektoru.
- kompozice.m - soubor funkce syntézy obrazu pomocí Framelet transformace.
- rozklad.m - soubor funkce analýzy obrazu pomocí Framelet transformace.
- sledovani_mysi.m - soubor funkce definování oblasti k doplnění, pomocí myši.
- threshold.m - soubor funkce prahování.
- vektor_na_koeficient.m - soubor pomocné funkce pro zpětné poskládání koeficientů Framelet transformace z vektoru zpět do původního rozložení.
- zobraz_koeficienty.m - soubor pomocné funkce, která úhledně poskládá koeficienty Framelet transformace do jednoho obrazu.

Text

- xkovac19_BP.pdf - elektronický text této práce.

Výsledky

- **contourlet_amy_lee_maska_01**
- **contourlet_amy_lee_maska_02**
- **contourlet_amy_lee_maska_03**
- **framelet_amy_lee_maska_01**
- **framelet_amy_lee_maska_02**
- **framelet_amy_lee_maska_02**